修 士 学 位 論 文

題 名

次世代シーケンス技術による多数遺伝子座を用いた
ショウジョウバエの分子系統解析（英文）

指導教授　　田村　浩一郎　　教授

平成　27年　1月　7日　　提出

首都大学東京大学院

理工学研究科　生命科学専攻

学修番号　13881305

氏　名　岩崎　祐磨

学位論文要旨（修士（理学））

論文著者名　岩崎　祐磨

論文題名：A multi-gene phylogenetic analysis of *Drosophila* species by next generation sequencing technology.
（邦題）：次世代シーケンス技術による多数遺伝子座を用いたショウジョウバエの分子系統解析（英文）

ショウジョウバエ属は 1,193 種が記載され多様性に富むが、その中でもショウジョウバエ亜属 717 種、シマショウジョウバエ亜属 348 種が多数を占める。ショウジョウバエ属の系統解析は、モデル生物であるキイロショウジョウバエ(*Drosophila melanogaster*)を中心に 12 種で全ゲノム配列が決定されており、これらをもとに進められている。12 種中 9 種がキイロショウジョウバエを含むシマショウジョウバエ亜属に含まれ、ショウジョウバエ亜属は多くの種を抱えるにも関わらず十分になされていのが現状である。特に *D. quinaria* section にはショウジョウバエ亜属の半数近い種が属すが、全ゲノム配列が決定された種が存在しない。更に先行研究においては、ミトコンドリア遺伝子などの数遺伝子座のみを用いた系統解析しか行われておらず、その信頼性は低い。そこで本研究では *D. quinaria* section に着目し、核ゲノム由来の単一コピー遺伝子を多数用いた解析を行うことを目的とした。

本研究では *D. quinaria* section から、代表的な種群を形成する 9 種（*D. funebris, D. angulalis, D. orientacea, D. sternopleuralis, D. tripunctata, D. albomicans, D. bizonata, D. guarani, D. cardini*)を対象にトランスクリプトーム解析を行った。遺伝子情報の少なさをカバーするため、次世代シーケンサー（Roche 454 GS junior）を用いて成虫の RNA-seq を行い、1 種につき 3,000 遺伝子座程度の cDNA 配列を決定した。得られた配列データについては、ショウジョウバエ 12 種の全ゲノムデータを基にデータベース検索を行い、1 種につき核由来オーソログ遺伝子を 600 遺伝子座程度同定した。それらの中から、全種で共有する 161 遺伝子座を用いて系統解析を行ったところ、新大陸産のショウジョウバエ（*D. tripunctata, D. guarani, D. cardini*)は単系統群になることがわかった。また、*D. quinaria* section の種群は 1,000～3,100 万年前に分岐した可能性が高いことがわかった。しかし、DNA を用いた解析とアミノ酸を用いた解析では、樹形内での新大陸産のショウジョウバエの位置が若干異なり、どちらも高いブートストラップ値とベイズ法の事後確率を示した。大規模なデータセットによる解析であっても方法によって樹形が変化する可能性が示され、大規模データ解析のための理論的解析法の必要性が示唆された。

A multi-gene phylogenetic analysis of *Drosophila* species by the next generation sequencing technology

Yuma Iwasaki

Department of Biological Sciences
Graduate School of Science and Engineering
Tokyo Metropolitan University

Abstract

There are as many species as 1,193 in the genus *Drosophila* including 717 species in the subgenus *Drosophila* and 348 species in the subgenus *Sophophora*. In the genus *Drosophila,* whole genome sequences were determined for 12 species including *D. melanogaster,* a model organism. In these species, three species belong to the subgenus *Drosophila* and nine species including *D. melanogaster* belong to the subgenus *Sophophora*. The phylogenetic relationships among species in the subgenus *Sophophora* have been well studied but those in the subgenus *Drosophila* have not been so. The *D. quinaria* section, one of two major sections in the subgenus *Drosophila,* consists of many species but genome sequence has not been determined for any species. Phylogenetic analyses of the *D. quinaria* section have been based on only a small number of genes including mitochondrial genes. Therefore, I focused on the *D. quinaria* section and studied the phylogenetic relationships among the species in this section using a large number of orthologous nuclear genes.

In this study, I performed transcriptome cDNA sequence analyses for nine species (*D. funebris, D. angularis, D. orientacea, D. sternopleuralis, D. tripunctata, D. albomicans, D. bizonata, D. guarani* and *D. cardini*) in the *D. quinaria* section. I determined cDNA sequences of approximately 1,000 – 3,000 genes for each species by the RNA-seq method using a Roche 454 GS Junior sequencer to conduct a large scale phylogenetic analysis. In those genes, I identified 300 – 1,000 orthologous genes in each species searching the 12 genome sequence database. Finally, I analyzed the phylogeny based on 161 orthologous genes shared by all the species

used. The result showed that three New World species (*D. tripunctata, D. guarani* and *D. cardini*) were clustered as a monophyletic clade and the species groups in the *D. quinaria* section diverged during the period of 10 to 31 MYA. However, the positons of the New World species and *D. funebris* were inconsistent between DNA and protein datasets with high bootstrap values and Bayesian posterior probabilities, indicating the need of further improvement of theoretical methods for phylogenetic analyses with a big sequence dataset.

Index

Introduction

Fruit fly species belonging to the genus *Drosophila* have been well studied especially in the field of genetics because of its rapid alternation of generations and ease of strain maintenance. Phylogenetic relationships among fruit fly species have been also well studied because there were many species including a canonical model organism species, *Drosophila melanogaster*. According to DrosoDB database (http://bioinfo.lowtem.hokudai.ac.jp/db/modules/stdb/) and NCBI taxonomy database (http://www.ncbi.nlm.nih.gov/guide/taxonomy/), 4,100 species are described in the family Drosophilidae. Within this family, there are 1,193 species in the genus *Drosophila,* which includes the subgenera *Drosophila* (717 species) and *Sophophora* (348 species). In the genus *Drosophila*, whole genome sequences have been determined for 12 species (Drosophila 12 Genomes Consortium, 2007), among which, 9 species (*D. melanogaster, D. simulans, D. sechellia, D. erecta, D. yakuba, D. ananassae, D. pseudoobscura, D. persimilis* and *D. willistoni*) belong to the subgenus *Sophophora,* and 3 species (*D. virilis, D. mojavensis* and *D. grimshawi*) belong to the subgenus *Drosophila*.

Many studies have been studied on the phylogenetic relationships among *Drosophila* species (e.g., Sturtevant, 1921 and 1942; Patterson and Stone, 1952; Okada, 1956 and 1989; Throckmorton, 1975; Wheeler, 1981 and 1986; Grimaldi, 1990). One of the most remarkable studies was accomplished by Throckmorton (1962, 1956 and 1966). He compared internal reproductive organs and egg shapes among many species. On the basis of these works together with ecology and

biogeography, consequently, he proposed a phylogenetic hypothesis among the species of the family Drosophilidae (Throckmorton, 1975). In his hypothesis, the genus *Drosophila* was not monophyletic, including many clades of other genera within the genus. His hypothesis has been widely accepted by many researchers because of the broad coverage of species, in spite of the lack of objective facts. Based on 217 external morphological characters of adult flies, Grimaldi (1990) reconstructed a phylogenetic tree by the maximum parsimony method. In his alternative hypothesis, the subgenus *Drosophila* was monophyletic with *Hawaiian Drosophila* species, and the genus *Scaptomyza* was placed outside the genus *Drosophila*. Accordingly, there have been two very different hypotheses for the phylogeny of *Drosophila* species.

Recently, large scale molecular phylogenetic analyses for *Drosophila* species have been conducted, i.e., phylogenetic analyses for 12 species based on whole genome sequences (Drosophila 12 Genomes Consortium, 2007), for 134 species in 10 genera based on their mitochondrial genomes (O' Grady and De Salle, 2008) and for 176 species in 12 genera based on a super-matrix analysis of 13 genes (Linde et al., 2008 and 2010). The results of these studies were roughly congruent with Throckmorton's hypothesis rather than Grimaldi's. Yet, these studies have not clarified the phylogenetic relationships among many species groups, especially in the subgenus *Drosophila.* For example, only three genome sequences have been determined for the species in the subgenus *Drosophila,* so that only mitochondrial and a few nuclear genes have been used for the phylogenetic analyses among these species. In the subgenus *Drosophila*, there are two major lineages, i.e., the *D. virilis* section including 296 species, and the *D. quinaria* section including more than 283

species (Hsu 1949). Although the *D. quinaria* section is a relatively large section, no genome sequence data has been available yet for any species in this section.

To solve the problems due to a small number of genes including paralogs in the previous works, I focused on the RNA-seq method (Mortazavi et al., 2008; Marioni et al., 2008; Nagalakshmi et al., 2008) and determined millions of transcriptome cDNA sequences by using a high-throughput DNA sequencer to obtain the nucleotide sequences of over 100 orthologous nuclear genes for a comprehensive phylogenetic analysis of the species belonging to the *D. quinaria* section in the genus *Drosophila*. In this study, I determined transcriptome cDNA sequences for eight representative species in the *D. quinaria* section, i.e., *D. funebris, D. angularis, D. orientacea, D. sternopleuralis, D. tripunctata, D. bizonata, D. cardini* and *D. guarani*, using a Roche 454 junior sequencer. Together with the transcriptome cDNA sequence of *D. albomicans* determined in the previous work in my laboratory, and the 12 genome sequences, I analyzed the phylogenetic relationships among the 21 species using 161 orthologous nuclear genes, and revisited Throckmorton's (1975) hypothesis.

Materials and Methods

Eight species in the *D. quinaria* section

I selected 8 *Drosophila* species (*D. funebris, D. sternopleuralis, D. bizonata , D. anguralis, D. orientacea, D. tripunctata, D. cardini* and *D. guarani*) belonging to the subgenus *Drosophila* for the transcriptome cDNA determination (Table 1). They represent major species groups in the *D. quinaria* section. All the species were living isofemale lines maintained at 20 °C under constant light condition in our laboratory. The strains of *D. funebris, D. sternopleuralis, D. bizonata, D. anguralis, D. orientacea* and *D. tripunctata* were originally collected in Japan, whereas *D. cardini* and *D. guarani* were obtained from the UC San Diego Drosophila Stock Center.

Extraction of total RNA with a silica gel method

Total RNA was isolated from 10 adult flies for each species, using a combined method of the extraction by acid guanidinium thiocyanate–phenol–chloroform (Chomczynski and Sacchi, 1987) and the purification by silica particles (Boom et al. 1999) as follows.

Ten adult flies were homogenized in 500 µl Binding Buffer (4 M Guanidine Thiocyanate; 25 mM Sodium Citrate; 0.5 % SDSc; 0.1 M Mercaptoethanol) and then 10 µl silica gel mixture (spherical shape of silica gel particles of 5 µm diameter suspended in equal volume of 0.01 N HCl) was added into the homogenate. After 5 minute incubation at room temperature with vortexing every 1 minute to absorb DNA on the silica gel particles, the homogenate was centrifuged at 12,000 rpm for 1 minute at 20 °C, and the supernatant was transferred into a new tube. Then, 50 µl

of 2 M Sodium Acetate (pH = 4.0) and 500 µl of water-saturated phenol were added. After vortexing the tube, 150 µl chloroform : isoamylalcohol (24 : 1) was added and the tube was vortexed again thoroughly and put on ice for 15 minutes. Then, the tube was centrifuged at 12,000 rpm for 20 minutes at 4 °C, and the supernatant was recovered to a new tube. A half volume of the supernatant (approximately 500 µl) of 100 % ethanol and 10 µl silica gel mixture were added into the tube and incubated at room temperature for 5 minutes with vortexing every 1 minute to absorb RNA on the silica gel particles. After centrifugation at 12,000 rpm for 1 minute at 20 °C, the supernatant was removed and the precipitate of silica particles was washed by 500 µl Wash Buffer (10mM Tris-HCl pH 7.5; 100mM NaCl : Ethanol = 1 : 4) three times. Then, the resultant precipitate of silica particles was air-dried with a blower to remove the residual ethanol. Finally, RNA absorbed on the silica particles was eluted with 50 µl TE (10mM Tris-HCl pH 8.0; 0.1mM EDTA) by incubation at 70°C for 5 minutes. The elution was centrifuged at 12,000 rpm for 1 minute at 20 °C and the supernatant was recovered into a new tube as a RNA sample. The quality of each RNA sample was checked by an Agilent 2100 Bioanalyzer and stored at -80°C until use.

Transcriptome cDNA sequencing

From 20 to 30 µg total RNA, mRNA was purified by Ambion Dynabeads® mRNA Purification Kit. Using 200ng mRNA obtained, I synthesized cDNA library according to the Roche GS Junior Titanium Series rapid library cDNA synthesis manual.

The emulsion PCR (emPCR) was conducted by using a Roche GS Junior

Titanium emPCR Kit (Lib-L) with the obtained cDNA library as the template at the concentration ratio of 10 : 1 to the capture beads. After the emPCR, I input the resultant beads into a Roche 454 GS junior sequencer and run it according to the instruction by the supplier (Roche, Ltd).

Assembling raw reads of DNA sequences

For the raw sequence reads output by the GS junior sequencer (about 7 to 15 thousands reads), I assembled them using GS DeNovo assembler (ver. 2.7) with default settings to produce 1,000 to 3,000 contigs. Next, using Blast search (Altschul et al., 1997) against the *D. melanogaster* database release 5.52 with the e-value equal to 1.0×10^{5}, I identified and removed the contigs from rDNA and mtDNA sequences. To analyze the Blast output files and to edit fasta sequence data files, I coded Java programs (Appendix). All the resultant contigs were then mapped to the translated amino acid sequences of *D. melanogaster* to obtain Flybase gene ID and other information using blastx program with the e-value equal to 1.0×10^{-5}. Using the information, I further corrected and assembled the sequence data by aligning to the 12 genome sequence alignment using Clustal Omega (http://www.clustal.org/). The work flow is illustrated in Figure 1.

Other sequence data

In addition to the sequence data newly determined in this study, I also utilized the cDNA sequence of *D. albomicans* previously determined in our laboratory and whole genome sequences for 12 species (*D. melanogaster, D. simulans, D. sechellia, D. erecta, D. yakuba, D. ananassae, D. pseudoobscura, D. persimilis, D. willistoni, D.*

virilis, *D. mojavensis* and *D. grimshawi*) downloaded from Flybase (ftp://ftp.flybase.net/releases/).

The orthology of each gene was checked by Flybase gene ID. As a result, I found that 161 orthologous genes were shared by all the 21 species. The nucleotide sequences of the 161 genes were aligned gene by gene according to their translated amino acid sequences using Clustal Omega (http://www.clustal.org/) with the default parameter setting. Finally, I concatenated the obtained sequence alignments for the 161 genes to be the dataset with 160,671 nucleotide sites and 53,557 amino acid sites. However, as the RNA-seq method does not necessarily cover the entire region of each gene sequence, there were many missing regions in the final alignment in addition to the real alignment gaps. Accordingly, the number of nucleotide sites of each species varied from 124,022 to 157,399 with the average of 151,951.5 among the nine species. To check the effects of the missing data, I built two subsets of data, i.e., the 90 % coverage dataset composed of the sites covered by 90 % or more species and the 100 % coverage dataset obtained by the 'complete deletion' option.

Phylogenetic analyses

The phylogenetic trees were reconstructed by the maximum likelihood (ML) method (Felsenstein 1981) and Bayesian method (Li, 1996; Mau, 1996). For the ML analysis, I used RAxML-standard SSE3-PTHREADS ver. 8.1.2 (Stamatakis, 2014) on Ubuntu 12.04 LTS with 100 bootstrap replications. The substitution models used were WAG (Whelan and Goldman, 2001) for amino acid sequences and GTR (Lanave et al., 1984) for nucleotide sequences. The gamma distribution option was used to fit the

rate heterogeneity among sites. For Bayesian analysis, I used MrBayes ver. 3.2.3 x64 MPI version (Robinquist et al., 2012) on Ubuntu 12.04 LTS. The parameter settings for the Markov chain Monte Carlo were the number of chains (nchains) = 4, the number of generations (ngen) = 20,000 and the sample frequency (samplefreq) = 100 for DNA datasets and nchains = 10, ngen = 10,000 and samplefreq = 500 for protein datasets. The substitution models were the same as those used in the ML analyses. Only these models were available in both programs.

In addition to the concatenated sequence alignment of the 161 genes, the ML analysis was conducted by the partitioned analysis with the WAG+GAMMA model with 100 bootstrap replications, where the values of gamma parameter and branch lengths were optimized gene by gene for each tree topology in the ML topology search. This analysis was done by using RAxML-standard SSE3-PTHREADS ver. 8.1.2 on Ubuntu 12.04 LTS.

To avoid false negative evaluation of alternative topologies in the bootstrap analyses, I used Shimodaira-Hasegawa test (Shimodaira et al., 1999) for protein and DNA datasets among three alternative topologies implemented in RAxML-standard SSE3-PTHREADS ver. 8.1.2.

Divergence time estimation

I computed the divergence times between the species groups in the *D. quinaria* section and others using Reltime (Tamura et al., 2012) on MEGA6 (Tamura et al., 2013). The divergence time of the subgenera *Drosophila* and *Sophophora* was assumed to be 62.9 MYA (Tamura et al., 2004) as the calibration point.

Results

Nucleotide sequences and assembled datasets

The Roche 454 GS Junior sequencer produced 80,000 - 150,000 raw reads of nucleotide sequence (Table 2). Among them, ribosomal DNA sequences and mitochondrial DNA sequences accounted for nearly 20 % and 7 % of the reads, respectively (Figure 2). The remaining raw reads were subjected to the assembling by GS de novo assembler to result in 1,000 - 3,000 contigs (Table 2). Approximately 70 % of the contigs obtained had homology to the nuclear protein coding genes of *D. melanogaster*, in which approximately 40 % were identified to be orthologous genes. Consequently, nucleotide sequences of 300 – 1,000 orthologous genes were obtained for each species (Table 2). For many of the genes, the nucleotide sequences were partial consisting of two or more contigs separated by missing regions. Compiling these sequences for the 21 species, I found that 174 genes were at least partially shared by all species, out of which 161 genes were aligned successfully whereas the alignment was failed for the remaining 13 genes. For the concatenated sequence data set, the numbers of nucleotide and amino acid sites were 160,671 and 53,557, respectively. The total number of nucleotide and amino acid sites varied among species: 124,022 – 157,399 nucleotide sites and 41,341 – 52,466 amino acid sites (Table 2).

Phylogenetic trees

The phylogenetic tree obtained from the concatenated nucleotide sequence alignment of the 161 genes is shown in Figure 3A. In this tree, 100 percent

bootstrap (BS) value and 1.0 Bayesian posterior probability (PP) were shown in all the interior branches. The three New World species (*D. cardini, D. guarani* and *D. tripunctata*) were clustered as a monophyletic clade and so as were the two species of the *immigrans* species group (*D. albomicans* and *D. sternopleuralis*). This tree indicates that, after the *D. quinaria* section and the *D. virilis* section diverged from each other, the *immigrans* species group diverged first and then the lineage to the three New World species and the lineage to the four Old World species (*D. funebris, D. angularis, D. orientacea* and *D. bizonata*) diverged. However, the translated amino acid sequences showed a different topology, where the New World lineage was replaced with *D. funebris* (Figure 3B). To investigate how this conflict occurred, I estimated the phylogeny using only the first and second codon positions and the third positions separately. The first and second codon positions showed the same topology as the amino acid sequences, whereas the third codon positions showed the same topology as the entire nucleotide sequences (Figure 4). However, the third codon positions caused longer branches within the subgenus *Sophophora* (Figure 4B).

To examine the influence of missing data, I estimated the phylogenetic tree using two data subsets. One was the 90 % coverage subset consisting of the 46,773 amino acid and 140,366 nucleotide sites that were covered by 90 % or more species (Figure 5A), and another was the 100 % coverage subset obtained by the 'complete deletion option' consisting of 33,885 amino acid and 101,907 nucleotide sites shared by all the species (Figure 5B). No topological difference was produced by either of the subsets. The bootstrap values for the 90 % coverage protein subset were higher than that for the 100 % coverage protein subset.

The Shimodaira-Hasegawa (SH) test (Shimodaira et al., 1999) was applied to three alternative tree topologies (Figure 6) constructed with the protein dataset by ML and Bayesian methods (topologies 1), with the DNA dataset by ML, Bayesian and NJ method (topology 2), and with protein dataset by ML and NJ methods (topology 3). As the result, the tree topology 1 was supported significantly better than the topologies 2 and 3 with the protein dataset, whereas the topology 2 was significantly better than the topologies 1 and 3 with the DNA dataset. Therefore, the results indicated that the same dataset supports the different topologies with statistical significance depending on whether it was analyzed at nucleotide or amino acid level.

The effect of GC content bias

To examine the effect of the variation of the G+C content among species, the G+C content of the 161 genes was computed for each species (Figure 7). There was a strong tendency that the G+C content was higher in the species of the subgenus *Sophophora* except *D. willistoni.* Another tendency was that the subgenus *Drosophila* had a higher variation among species compared to the subgenus *Sophophora* except *D. willistoni.* The difference between the subgenera and the variation among species were mainly attributed to the third codon positions (Figure 8).

To examine the influence of the G+C content variation among species, I classified the 161 genes by the variance of G+C content among species into two data subsets. One was the subset consisting of top 80 genes with low G+C content variation [standard deviation of GC% ($SD_{GC\%}$) was 0.50 to 3.17] among species and

another was the subset consisting of the remaining genes with higher variation ($SD_{GC\%}$ was 3.17 to 7.11). However, they produced the same topology as did the original DNA dataset (Figure 9). If I used only the top 20 genes ($SD_{GC\%}$ was 0.50 to 2.39) and bottom 20 genes ($SD_{GC\%}$ was 4.21 to 7.11), they produced almost the same result with a small difference as when top 20 genes were used (Figure 10).

Divergence time of the *D. quinaria* section

Figure 11 shows the divergence times for the *Drosophila* species estimated under the assumption that the subgenera *Drosophila* and *Sophophora* diverged 62.9 MYA (Tamura et al., 2004). I adopted the tree topology estimated from the protein dataset because the DNA dataset suggested different topologies among different codon positions. The result showed that the species groups in the *D. quinaria* section diverged during the periods of 10 to 31 MYA, in which the lineage of the New World species (*D. cardini, D. guarani* and *D. tripunctata*) diverged from the oriental species 19 MYA.

Discussion

Phylogenetic relationships in the *D. quinaria* section

In this study, two alternative phylogenetic relationships among the species in the *D. quinaria* section were obtained depending on whether the nucleotide or amino acid sequence data was used for the tree estimation. Although the three New World species were clustered as a monophyletic clade in both topologies, the position of the clade was different between the two topologies. The DNA dataset suggested that the New World clade was the sister group of four Asian species (*D. funebris, D. angularis, D. orientacea* and *D. bizonata*), whereas the protein dataset suggested that *D. funebris* and *D. angularis* diverged earlier than the divergence between the New World clade and *D. orientacea* and *D. bizonata* (Figures 3A and 3B). It was speculated by Throckmorton (1975) that the ancestor of the family Drosophilidae lived in tropical Asia and then expanded the distribution to all over the world. The obtained monophyly of the New World species in either of two topologies is consistent with this thought. Another consistent result is the close relationships between *D. sternopleuralis* belonging to the *D. histrio* species group and *D. albomicans* belonging to the *D. immigrans* species group. *D. sternopleuralis* was classified into the *D. histrio* species group by means of its external morphology, but the genital organs were not so similar to that of *D. histrio* (Okada, 1956). Therefore, there is a possibility that *D. sternopleuralis* belongs to other species group. To clarify whether these species groups are closely related or just *D. sternopleuralis* and *D. albomicans* are closely related, it is necessary to include the data of representative species of these species groups, i.e., *D. immigrans* and *D. histrio*.

In general, amino acid substitution is more conservative due to the functional constraints at protein level than nucleotide substitution that involves selectively neutral synonymous substitution. Because the higher substitution rate at synonymous sites often causes 'saturation effect' that potentially distorts the estimation of phylogenetic relationships, amino acid sequences are expected to be less error-prone (Jeffroy et al., 2006; Rota-Stabelli et al., 2010) except for the genes involved in immune systems (Foster and Hickey 1999). In the present case, the phylogenetic tree based on the third codon positions showed imbalanced branch lengths with long branches among the species in the subgenus *Sophophora,* which may be a sign of distortion in analyzing the DNA dataset. These branch lengths were negatively correlated with G+C contents; the top 20 low variation genes showed a slightly different topology with longer branch lengths than the top 20 high variance genes (Figure 10). There seems to be heterotachy (Lopez et al. 2002) caused by the G+C content bias at third codon positions. Nevertheless, I did not find any evidence in this study that the bias had an effect on the topology estimation.

Previous studies on *Drosophila* phylogeny was performed at the DNA level (e.g., van der Linde et al., 2010). This may be because nucleotide sequences are three times longer than their translated amino acid sequences, which has an advantage to reduce the sampling error to obtain higher bootstrap values and Bayesian posterior probabilities. In this study, however, the total sequence lengths were long enough to use translated amino acid sequences as suggested by the high BS values and PP in the estimated tree. Therefore, I think that the tree topology estimated by using the protein dataset is more reliable in this study.

So far, most of the molecular phylogenetic studies were roughly congruent

with Throckmorton's (1975) hypothesis. In his hypothesis, the genera *Scaptomyza*, *Hirtodrosophila* and *Zaprionus* were placed in the *D. quinaria* section, suggesting that the genus *Drosophila* is not monophyletic including these other genera. It is worth examining this paraphyletic situation in the genus *Drosophila* by using a genome scale dataset.

Divergence time of *D. quinaria* section

After the *D. quinaria* section diverged from the *D. virilis* section 45 MYA, the species in the *D. quinaria* section diverged during the periods of 31 - 10 MYA on the basis of the protein dataset (Figure 11). The common ancestor of the *D. quinaria* section likely lived in Asian temperate zones, because currently all species I analyzed live in the Asian temperate zones. The New World lineage diverged 19 MYA in the mid Miocene era. If so, the ancestor of the New World species should migrate from Asian temperate zones to the New World in relatively warm climate in Miocene over Bering Strait (Figure 12). However, based on the DNA dataset, the ancestor of the New World species diverged 30 MYA in the mid Oligocene era. According to Throckmorton's (1975) hypothesis, yet this estimate agrees with the earlier migration event from Asian tropical zone to the New World tropical zone. Since there were so many species in Caribbean tropical zones, the ancestor of the New World tropical species was hypothesized to migrate from Asia and then extended their distribution to the New World temperate zone. Further studies including more species with genome scale data is required to conduct more accurate divergence time estimation.

Conclusion

In this study, focusing on the phylogenetic relationships among species in the *D. quinaria* section, I determined transcriptome cDNA sequences for eight *Drosophila* species and conducted phylogenetic analyses together with the already available transcriptome and genome sequence data. This analysis suggested two topologies with much higher reliability compared to previous works. It is expected that the discrepancy between the DNA and protein datasets could not be solved by using more genes. Novel theoretical innovations may help phylogenetic analyses with a big sequence dataset produced by high-throughput DNA sequencers in the nearest future.

In this study, it was clearly shown that the New World species, *D. tripunctata, D. cardini* and *D. guarani*, were clustered as a monophyletic clade, suggesting that they have a common ancestor migrated from Asia. The divergence time estimation suggested that the species groups in the *D. quinaria* section diverged during the periods of 10 - 31 MYA in the mid Oligocene era and the New World species diverged 19 MYA in the mid Miocene era. These timings are consistent with Throckmorton's (1975) hypothesis for the evolution of the family Drosophilidae.

Acknowledgement

I would like to express my deepest gratitude to Dr. Koichiro Tamura for his guidance during the course of this study and for critical reading of the manuscript. I am also deeply grateful to Dr. Yosuke Seto for his guidance of treatments of 454 GS Junior sequencer and its sequence data. I received samples of *D. guarani* and *D. cardini* from Drosophila Stocks of Ehime University so I owe a very important debt to Dr. Masayoshi Watada. I also thank all the members of the Evolutionary Genetics Laboratory for their kind help and invaluable discussions.

Reference

Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W and Lipman DJ, 1997, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucleic Acids Res 25: 3389-3402.

Blakey R, Wieneke U and Stoutjesdijk H, 2008, Map of the Miocene (20 ma), *In*: Wieneke U, Stoutjesdijk H, Simonet P, Liverani V (eds), *Gastropoda Stromboidea*. modified: June 13 2008 at 21:25; URL: http://www.stromboidea.de/?n=People.RonBlakey (accessed: Feb 17, 2015, at 20:00).

Bond JE, Garrison NL, Hamilton CA, Godwin RL, Hedin M and Agnarsson I, 2014, Phylogenomics resolves a spider backbone phylogeny and rejects a prevailing paradigm for orb web evolution, Curr Biol 24: 1765-1771.

Boom R, Sol C, Beld M, Weel J, Goudsmit J and Wertheim-van Dillen P, 1999, Improved silica-guanidiniumthiocyanate DNA isolation procedure based on selective binding of bovine alpha-Casein to silica particles, J Clin Microbiol 37: 615-619.

Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K and Madden TL, 2009, BLAST+: architecture and applications, BMC Bioinformatics 10: 421.

Chomczynski P and Sacchi N, 1987, Single-step method of RNA isolation by acid guanidinium thiocyanate-phenol-chloroform extraction., Anal Biochem 162: 156-159.

Drosophila 12 Genomes Consortium, Clark AG, Eisen MB, Smith DR, Bergman CM, Oliver B, Markow TA, Kaufman TC, Kellis M, Gelbart W and Iyer VN et al., 2007, Evolution of genes and genomes on the *Drosophila* phylogeny, Nature 450: 203-218.

Felsenstein J, 1981, Evolutionary trees from DNA sequences: A maximum likelihood approach, J Mol Evol 17: 368-376.

Foster PG and Hickey DA, 1999, Compositional bias may affect both DNA-based and protein-basedphylogenetic reconstructions, J Mol Evol 48: 284-290.

Grimaldi DA, 1990, A phylogenetic, revised classification of genera in the Drosophilidae (Diptera), Bull Am Mus Nat Hist 197: 1-139.

Hsu TC, 1949, The external genital apparantus of male Drosophilidae in relation to systematics, Univ Texas Publ 4920: 80-142.

Jeffroy O, Brinkmann H, Delsuc F and Philippe H, 2006, Phylogenomics: the beginning of incongruence?, Trends Genet 22: 225-231.

Lanave C, Preparata G, Saccone C and Serio G, 1984, A new method for calculating

evolutionary substitution rates, J Mol Evol 20: 86-93.

Li S, 1996, Phylogenetic tree construction using Markov chain Monte Carlo, PhD Dissertation, Ohio State Univ, Columbus.

Lopez P, Casane D and Philippe H, 2002, Heterotachy, an important process of protein evolution, Mol Biol Evol 19: 1-7.

Marioni JC, Mason CE, Mane SM, Stephens M and Gilad Y, 2008, RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays, Genome Res 18: 1509-1517.

Mau B, 1996, Bayesian phylogenetic inference via Markov chain Monte Carlo methods, PhD Dissertation, Univ of Wisconsin, Madison.

Mortazavi A, Williams BA, McCue K, Schaeffer L and Wold B, 2008, Mapping and quantifying mammalian transcriptomes by RNA-Seq, Nat Methods 5: 621-628.

Nagalakshmi U, Wang Z, Waern K, Shou C, Raha D, Gerstein M and Snyder M,

2008, The transcriptional landscape of the yeast genome defined by RNA sequencing, Science 320: 1344-1349.

Okada T, 1956, Systematic study of Drosophilidae and allied families of Japan, Gihodo, Tokyo.

Okada T, 1989, A proposal of establishing tribes for the family Drosophilidae with key to tribes and genera (Diptera), Zool Sci 6: 391-399.

O'Grady P and Desalle R, 2008, Out of Hawaii:the origin and biogeography of the genus *Scaptomyza* (Diptera: Drosophilidae), Biol Lett 4:195-199.

Patterson JT and Stone WS, 1952, Evolution in the genus *Drosophila*, MacMillan, NewYork.

Ronquist F and Huelsenbeck JP, 2003, MRBAYES3: Bayesian phylogenetic inference under mixed models, Bioinformatics 19: 1572-1574.

Rota-Stabelli O, Kayal E, Gleeson D, Daub J, Boore JL, Telford MJ, Pisani D, Blaxter M and Lavrov DV, 2010, Ecdysozoan mitogenomics: evidence for a common origin of the legged invertebrates, the Panarthropoda, Genome Biol Evol 2: 425-440.

Shimodaira H and Hasegawa M, 2001, CONSEL: for assessing the confidence of

phylogenetic tree selection, Bioinformatics 17: 1246-1247.

Siebert PD and Chenchik A, 1993, Modified acid guanidinium thiocyanate- phenol -chloroform RNA extraction method which greatly reduces DNA contamination, Nucleic Acids Res 21: 2019-2020.

Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD and Higgins DG, 2011, Fast, scalable generation of high - quality protein multiple sequence alignments using Clustal Omega, Mol Syst Biol 7: 539.

Stamatakis A, 2014, RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies, Bioinformatics 30: 1312-1313.

Sturtevant AH, 1921, The North American species of *Drosophila*, Carnegie Instituttion of Washington, Washington.

Sturtevant AH, 1942, The classification of the genus *Drosophila*, with descriptions of nine new species, Univ Texas Publ, 4213: 5-51.

Tamura K, Battistuzzi FU, Billing-Ross P, Murillo O, Filipski A and Kumar S, 2012, Estimating divergence times in large molecular phylogenies, Proc Natl Acad Sci USA 109: 19333-19338.

Tamura K, Stecher G, Peterson D, Filipski A and Kumar S, MEGA6: Molecular Evolutionary Genetics Analysis Version 6.0, Mol Biol Evol 30: 2725-2729.

Throckmorton LH, 1962, The problem of phylogeny in the genus *Drosophila*, Univ Texas Publ 6205: 207-343.

Throckmorton LH, 1965, Similarity versus relationship in *Drosophila*, Syst Zool 14: 221-236.

Throckmorton LH, 1966, The relationships of endemic Hawaiian Drosophilidae, Univ Texas Publ 6615: 335- 396.

Throckmorton LH, 1968, Biochemistory and taxonomy, Annu Rev Entomol 13: 99-114.

Throckmorton LH, 1975, The phylogeny, ecology and geography of *Drosophila*, *In*: King RC (ed), Handbook of Genetics 3: 421-469, Plenum Press, New York.

Throckmorton LH, 1977, Drosophila systematics and biochemical evolution, Annu Rev Ecol Syst 8: 235-254.

van der Linde K, Houle D, Spicer GS and Steppan SJ, 2010, A supermatrix-based molecular phylogeny of the family Drosophilidae, Genet Res 92: 25-38.

van der Linde K and Houle D, 2008, A supertree analysis and literature review of the genus Drosophila and closely related genera (Diptera, Drosophilidae), Insect Syst Evol 39: 241-267.

Wheeler MR, 1981, The Drosophilidae: a taxonomic overview, *In*: Ashburner M, Carson HL and Thompson JN (eds), The Genetics and Biology of Drosophila 3a: 1-97, Academic Press, New York.

Wheeler MR, 1986, Additions to the catalog of the world's Drosophilidae, *In*: Ashburner M, Carson HL and Thompson JN (eds), The Genetics and Biology of Drosophila 3e: 395-409, Academic Press, New York.

Whelan S and Goldman N, 2001, A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach, Mol Biol Evol 18: 691-699.

Tables

Table 1. *Drosophila* species used in this study and their subgenus, species group and distribution

Species	Subgenus	Species group	Distribution
Drosophila albomicans	*Drosophila*	*immigrans*	Asia
Drosophila angularis	*Drosophila*	*quinaria*	Asia
Drosophila bizonata	*Drosophila*	*bizonata*	Asia and Hawaii
Drosophila cardini	*Drosophila*	*cardini*	N. America
Drosophila funebris	*Drosophila*	*funebris*	Cosmopolitan
Drosophila guarani	*Drosophila*	*guarani*	S. America
Drosophila orientacea	*Drosophila*	*testacea*	Asia
Drosophila sternopleuralis	*Drosophila*	*histrio*	Asia
Drosophila tripunctata	*Drosophila*	*tripunctata*	N. America

Table 2. Information on sequence reads and assembled contigs

species	No. of raw reads	No. of all contigs	rDNA	mtDNA	Others	No. of Orthologous genes	No. of sites in 161 genes
D. albomicans		11,358	13	2	10,256	4,001	156,839
D. angularis	142,957	2,750	7	17	2,111	891	151,401
D. bizonata	155,612	2,695	10	20	2,041	844	150,136
D. cardini	148,526	3,240	18	25	2,582	1,082	151,243
D. funebris	154,851	2,702	14	25	1,913	719	149,124
D. guarani	147,422	2,656	16	11	1,970	799	150,890
D. orientacea	156,497	3,092	3	14	2,328	972	149,461
D. sternopleuralis	139,435	2,552	11	12	1,858	693	141,176
D. tripunctata	80,432	1,249	9	17	938	319	124,022
D. ananassae	—	—	—	—	—	6,675	156,132
D. erecta	—	—	—	—	—	6,675	157,334
D. grimshawi	—	—	—	—	—	6,675	156,071
D. melanogaster	—	—	—	—	—	6,675	157,399
D. mojavensis	—	—	—	—	—	6,675	156,888
D. persimilis	—	—	—	—	—	6,675	154,409
D. pseudoobscura	—	—	—	—	—	6,675	156,743
D. sechellia	—	—	—	—	—	6,675	156,117
D. simulans	—	—	—	—	—	6,675	146,004
D. virilis	—	—	—	—	—	6,675	155,724
D. willistoni	—	—	—	—	—	6,675	156,797
D. yakuba	—	—	—	—	—	6,675	157,071

Figures

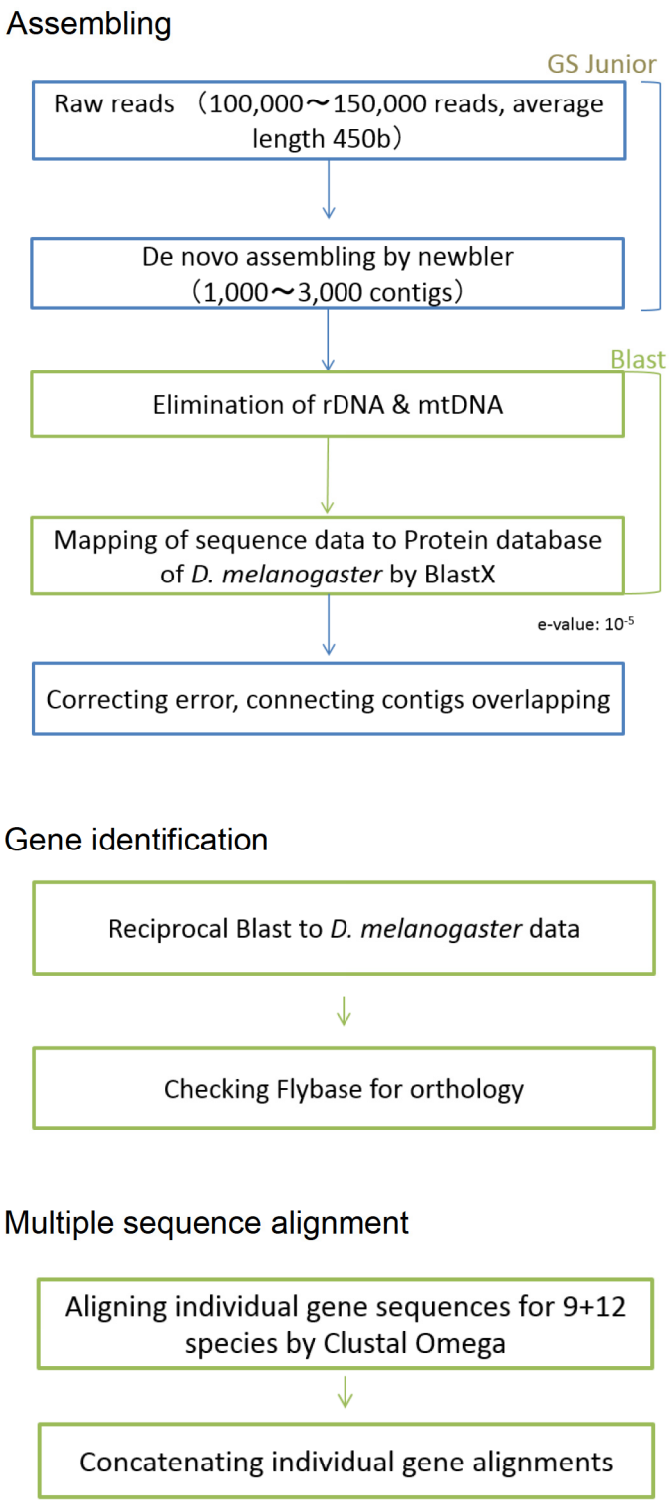


Figure 1. Work flow of building datasets.

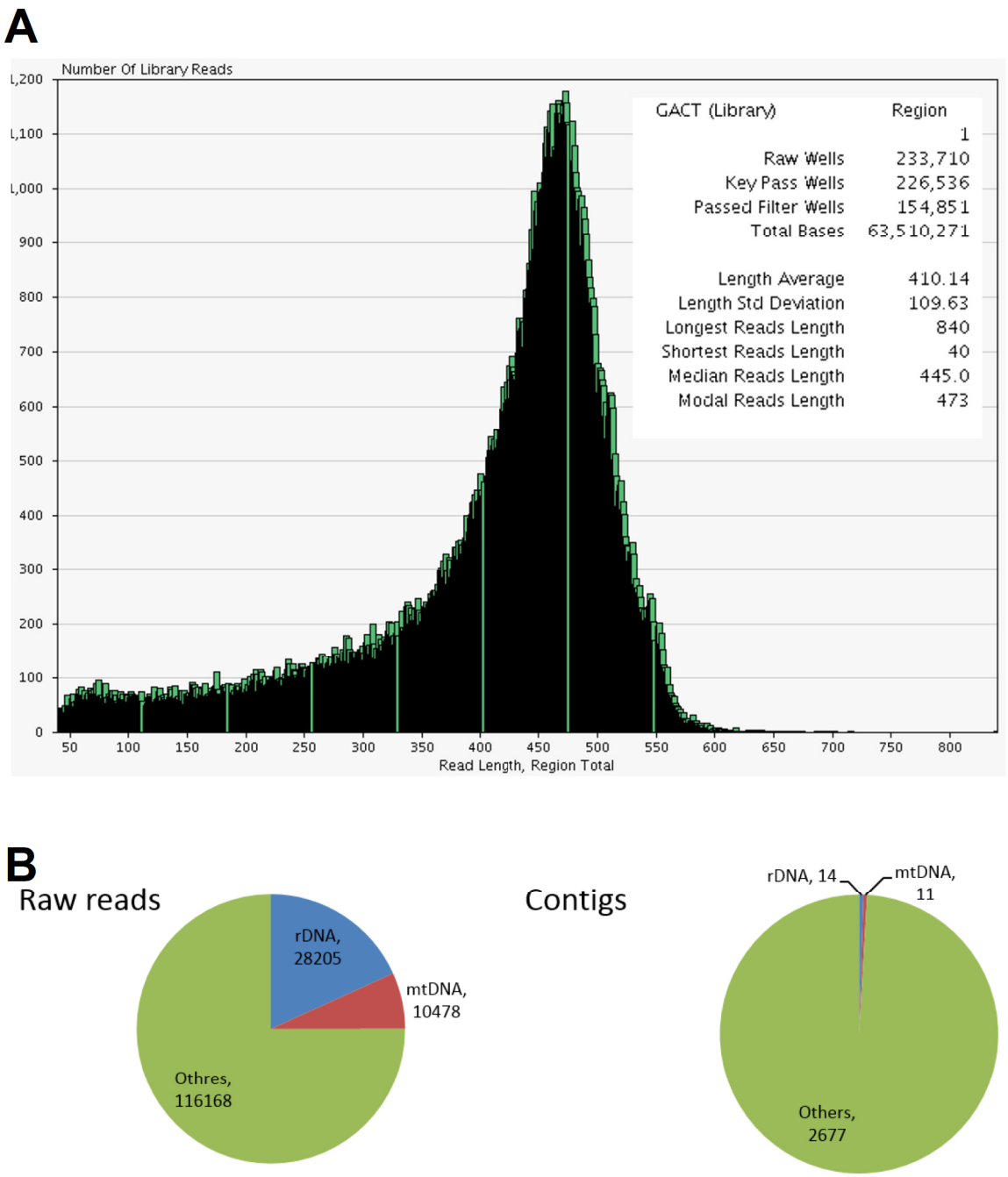


Figure 2. Example of raw sequence data obtained for *D. funebris.* A: Histogram of raw read lengths. B: Pie chart for the numbers of raw reads and assembled contigs resulting from ribosomal, mitochondrial and other DNAs.

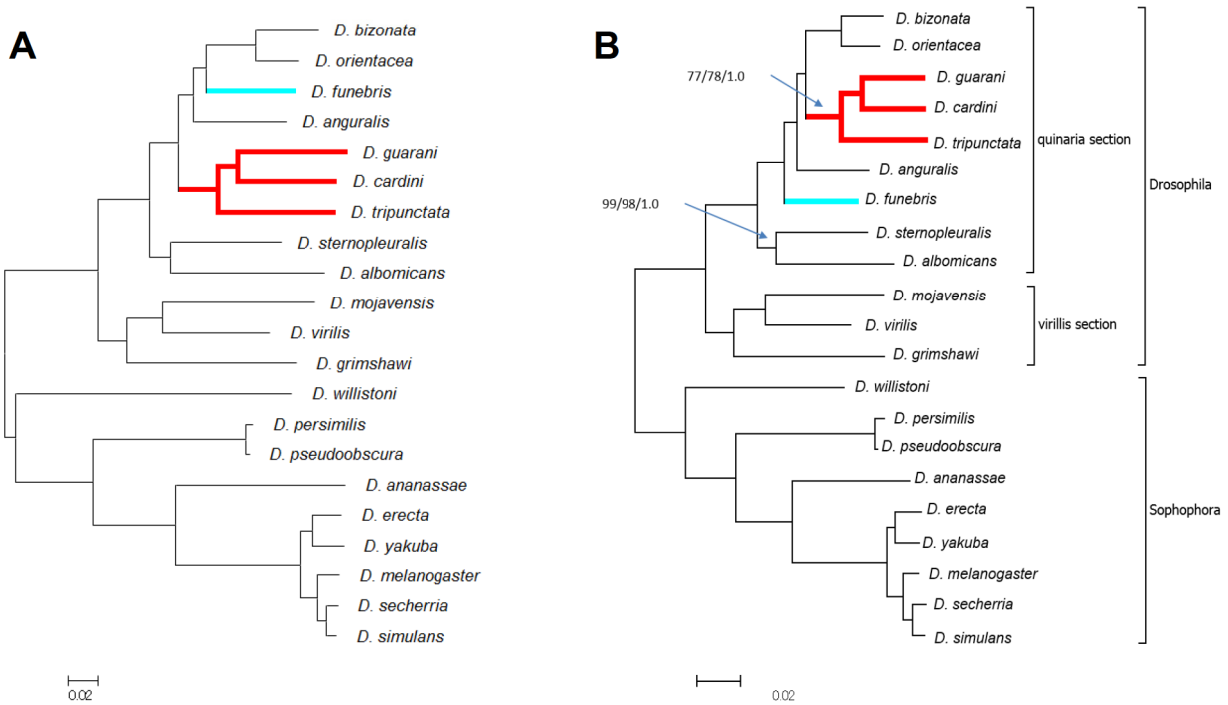


Figure 3. Phylogenetic trees obtained from nucleotide sequences (A) and from amino acid sequences (B). All interior branches other than two pointed by arrows showed 100 % bootstrap values by the ML method, 100 % bootstrap values by the partitioned ML method and 1.0 Bayesian posterior probabilities. The New World lineages are marked in red and the lineage to *D. funebris* are marked in cyan.

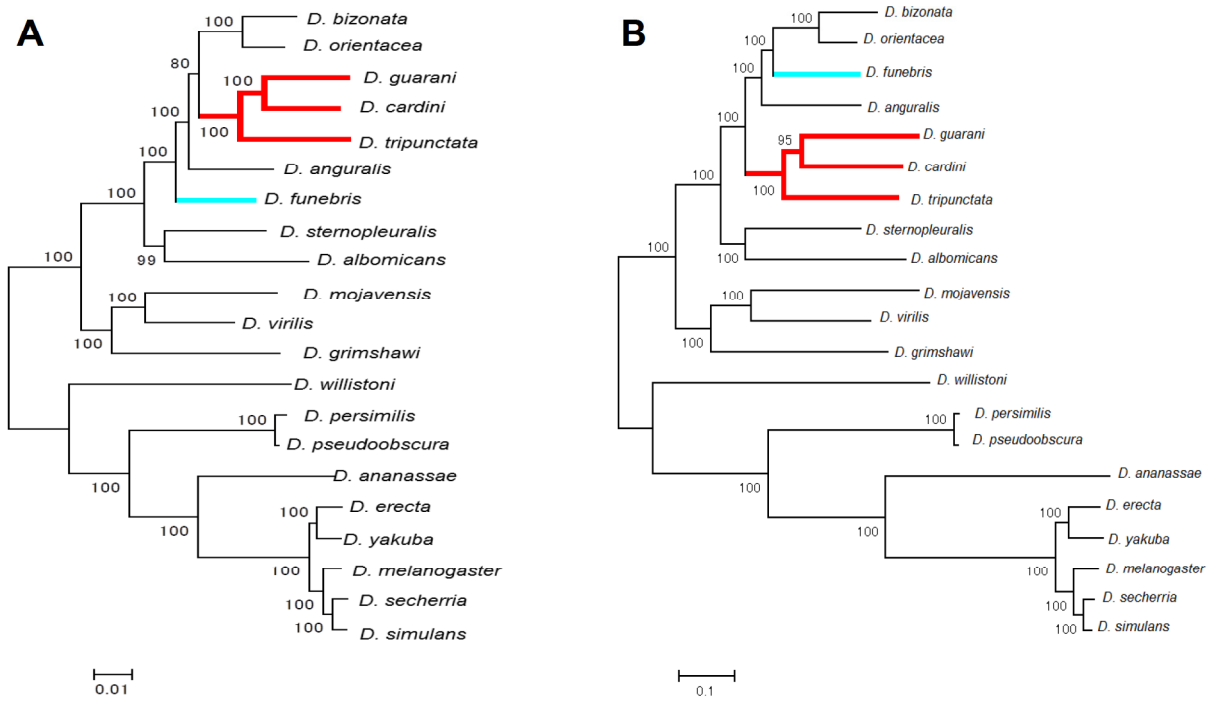


Figure 4. Phylogenetic trees obtained from first and second codon positions (A) and third codon positions (B). The numbers among interior branches are the bootstrap values. The New World lineages are marked in red and the lineage to *D. funebris* are marked in cyan.

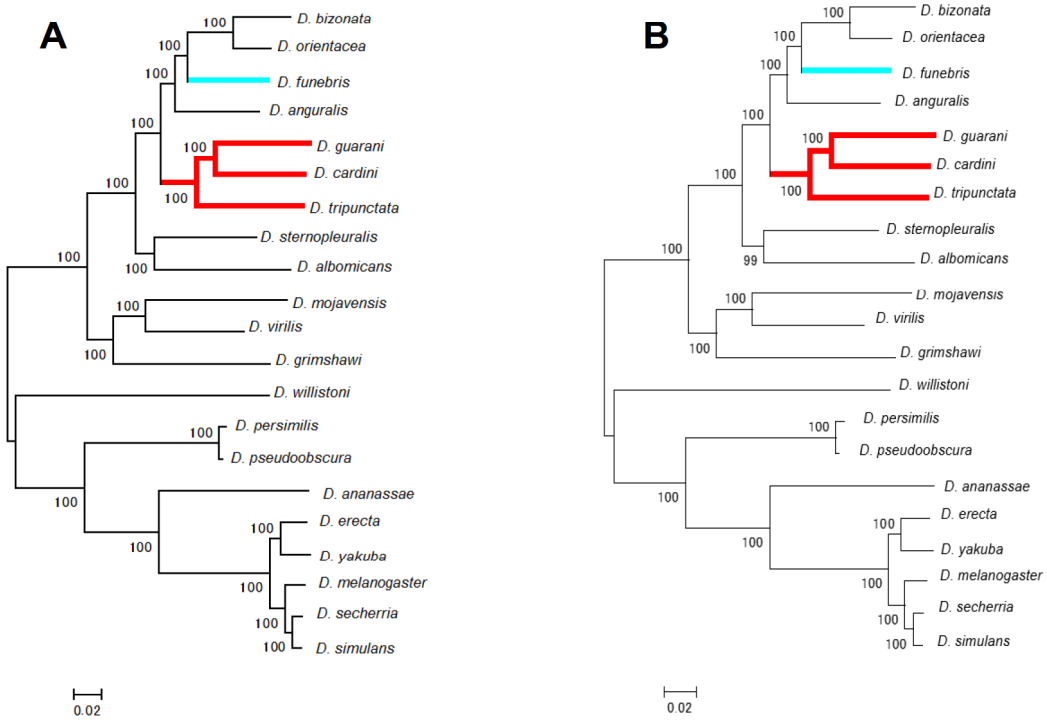
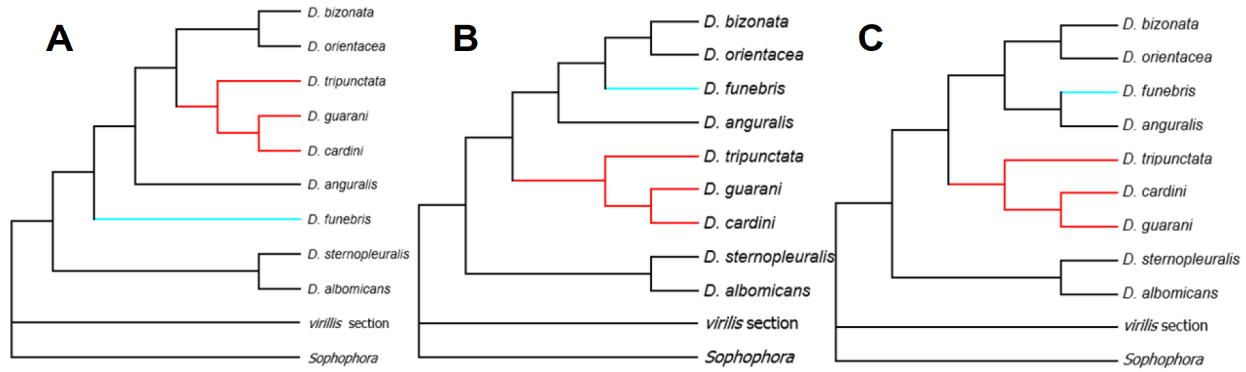


Figure 5. Phylogenetic trees based on 90 % (A) and 100 % (B) coverage DNA sites. The numbers among interior branches are the bootstrap values. The New World lineages are marked in red and the lineage to *D. funebris* are marked in cyan.



Topology 1 Topology 2 Topology 3

Protein dataset

Topology	log likelihood	△log likelihood	Standard Deviation	p-value
1	-380325.4	—	—	—
2	-380501.5	-176.1	52.1	< 0.01
3	-380477.8	-152.4	50.2	< 0.01

DNA dataset

Topology	log likelihood	△log likelihood	Standard Deviation	p-value
1	-1149780.6	-445.1	80.8	< 0.01
2	-1149335.5	—	—	—
3	-1149542.2	-206.7	40.5	< 0.01

Figure 6. Three alternative topologies and their test statistics with the results of Simodaira-Hasegawa test.
Topology 1 was constructed by the protein dataset with ML and Bayesian methods, topology 2 was constructed by the DNA dataset with ML, Bayesian and NJ methods and topology 3 was constructed by the protein dataset with ML and NJ methods. The lineages to the New World species are marked in red and the lineage to *D. funebris* are marked in cyan.

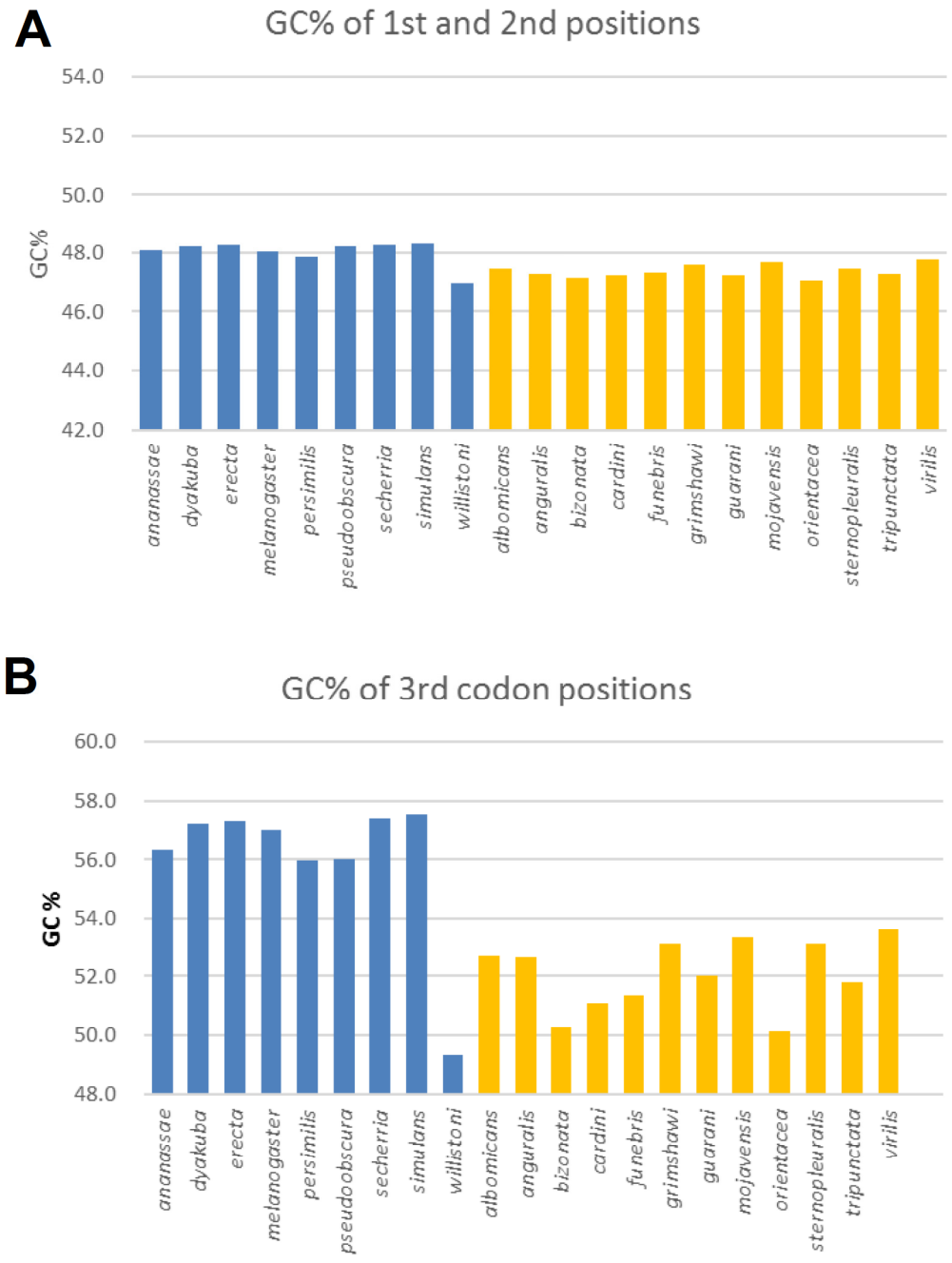


Figure 7. Histograms of GC% among species at first and second codon positions (A) and third codon positions (B). The species belonging to the subgenus *Sophophora* and *Drosophila* are indicated by blue and yellow colors, respectively.

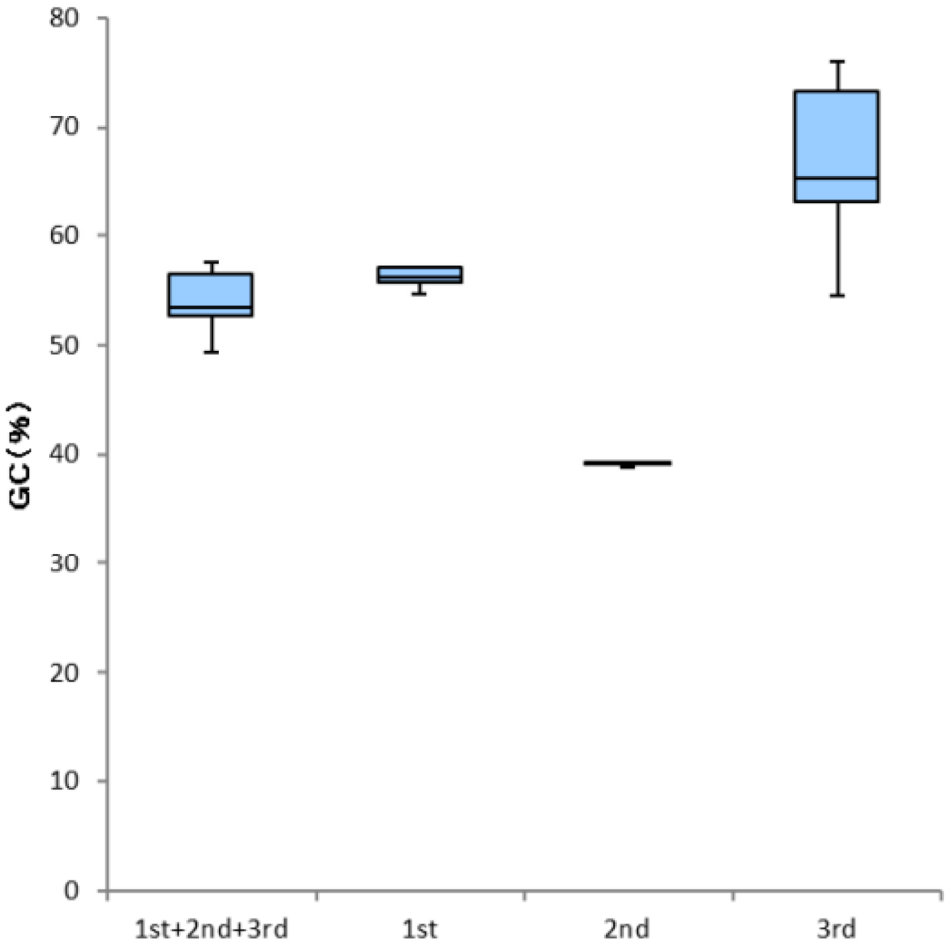


Figure 8. Box-and-whisker diagram of GC% by codon position.

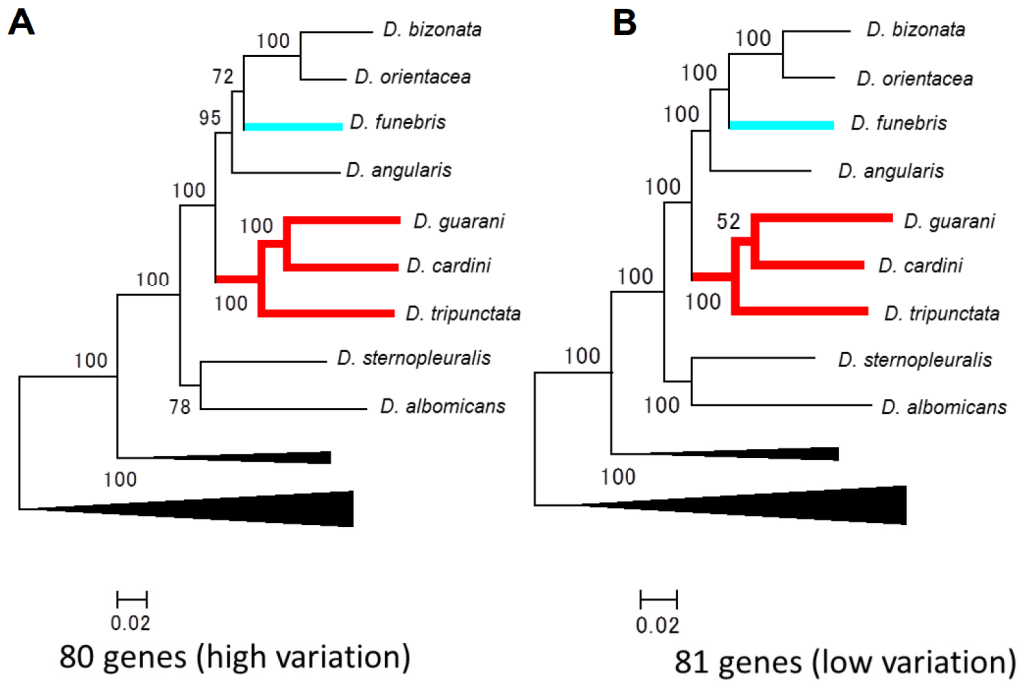


Figure 9. Phylogenetic trees based on the genes with high (A) and low (B) GC% variation among species. The numbers among interior branches are the bootstrap values. The lineages to the New World species are marked in red and the lineage to *D. funebris* are marked in cyan.

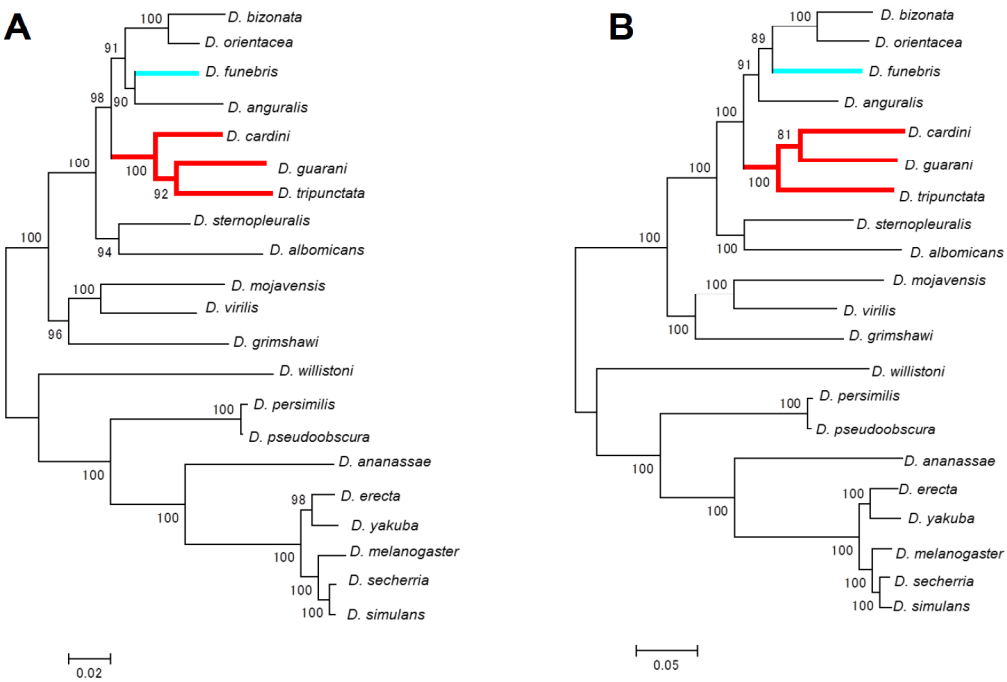


Figure 10. Phylogenetic trees based on 20 low (A) and 20 high (B) GC% variation genes. The numbers among interior branches are the bootstrap values. The lineages to the New World species are marked in red and the lineage to *D. funebris* are marked in cyan.

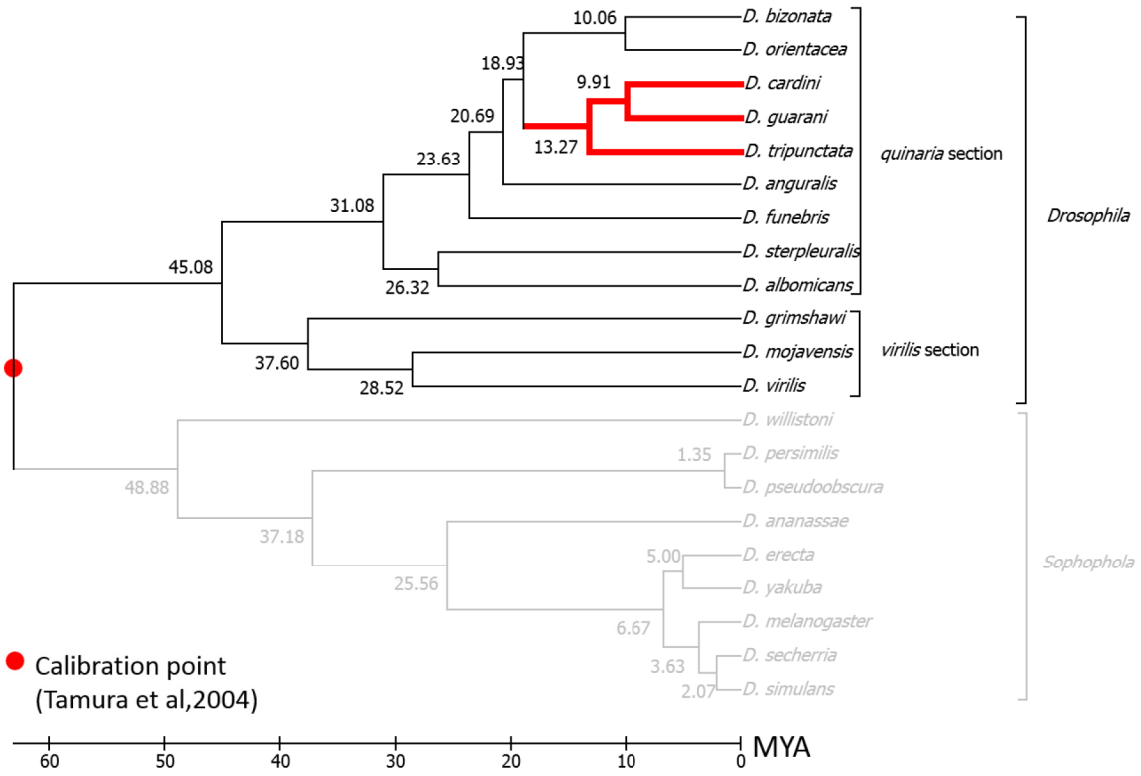


Figure 11. Estimated divergence times for the subgenus *Drosophila* species. The lineages to the New World species are marked in red.

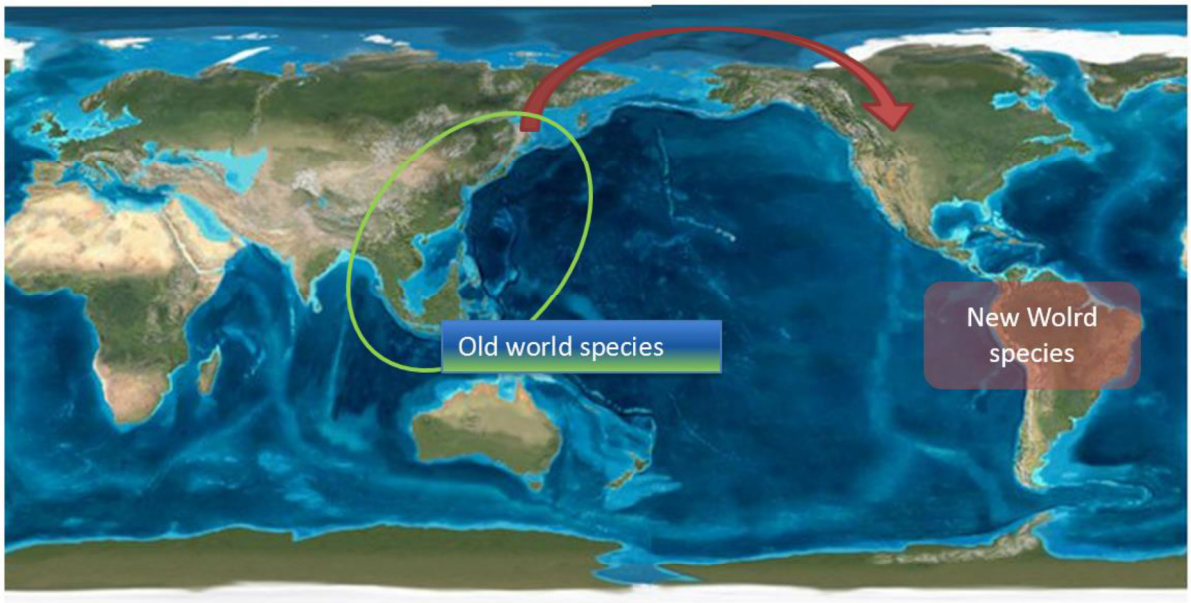


Figure 12. Potential migration route of the New World species from the Old World. The world map in Miocene era was according to Blackey et al. (2008). The common ancestor of the New World species lived in the Old World migrated through the Bering Strait to the New World in Miocene.

Appendix

Operating environment

OS: windows 7 Ultimate 64 bit

Integrated Developmental Environment:

Eclipse Java EE IDE for Web Developers.

Version: Kepler Service Release 2

Build id: 20140224-0627

External libraries

Biojava version 4.0.0 (http://biojava.org/wiki/Main_Page)

Apache POI - the Java API for Microsoft Documents version 3.1.1 (http://poi.apache.org/)

I coded all class files in a same package folder. Almost all softwares were GUI, developed by Swing Designer 1.6.1.r43x201309100023.

Output files Reader of Blast+ & Fasta file editor

```
package practice;
import javax.swing.*;
import javax.swing.text.DefaultEditorKit.*;

import org.biojava3.core.sequence.*;
import org.biojava3.core.sequence.storage.BitSequenceReader;
import org.biojava3.data.sequence.*;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


@SuppressWarnings("serial")
public class NGSfileEditor extends JFrame implements ActionListener{
		private JTextArea textArea = new JTextArea();
		//カット、コピー、ペースト用アクション
		private Action cutAction = new CutAction();
		private Action copyAction = new CopyAction();
		private Action pasteAction = new PasteAction();

		private JButton open = new JButton("outFile を開く");
		private JButton openFasta = new JButton("fasta を開く");
		private JButton save = new JButton("保存");
		private JButton cut = new JButton(cutAction);
		private JButton copy = new JButton(copyAction);
		private JButton paste = new JButton(pasteAction);
		private JButton runOutFile = new JButton ("runOutFile");
		private JButton removeHit = new JButton("Hit 配列を除いて保存");
		private JButton saveHit = new JButton("Hit 配列のみを取り出す");

		private JFileChooser chooser = new JFileChooser("C:¥¥cDNAData");
		private String fileAdress;
		private ArrayList seqlist;
		private ArrayList<ArrayList> indelList;//ヒットエリアリストのリスト、seqlist と同じ数だけ存在
		private ArrayList<IndelData> hitareaList;//各配列内のインデルリスト
		private List<FastaSequence> fastaList;
		private ArrayList<FastaSequence> fastaList2 = new ArrayList<FastaSequence>();

		//コンストラクタ
		public NGSfileEditor(){
			super("LocalBlast 結果解析");
			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			setSize(800,600);


			JPanel panel1 = new JPanel();
			JPanel panel2 = new JPanel();
			getContentPane().add(panel1,BorderLayout.NORTH);
			getContentPane().add(new JScrollPane(textArea),BorderLayout.CENTER);
			getContentPane().add(panel2,BorderLayout.SOUTH);

			panel1.add(open);
			panel1.add(openFasta);
			panel1.add(save);
			panel1.add(runOutFile);
			panel1.add(removeHit);
			panel1.add(saveHit);
			panel2.add(cut);
			panel2.add(copy);
			panel2.add(paste);

			open.addActionListener(this);
			openFasta.addActionListener(this);
			save.addActionListener(this);
			runOutFile.addActionListener(this);
			removeHit.addActionListener(this);
			saveHit.addActionListener(this);

			chooser.setAcceptAllFileFilterUsed(false);

		}

		public static void main(String[] args) {
			new NGSfileEditor().setVisible(true);
		}

		//[開く][保存]ボタンが押された時の処理
		public void actionPerformed(ActionEvent event){
			if (event.getSource().equals(open)){
				try {
```

```
                                        openOut();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                        else if (event.getSource().equals(openFasta)) {
                                openFasta();
                        }
                        else if (event.getSource().equals(save)){
                                saveFile();
                        }
                        else if (event.getSource().equals(runOutFile)){
                                try {
                                        runOutFile();
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                        else if (event.getSource().equals(removeHit)){
                                try {
                                        removeHit();
                                } catch (FileNotFoundException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                        else if (event.getSource().equals(saveHit)){
                                try {
                                        saveHit();
                                } catch (FileNotFoundException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
        }

        //ファイルを開くメソッド
        private void openOut() throws FileNotFoundException, IOException {
                        chooser.setFileFilter(new OutFilter());
                        int returnVal = chooser.showOpenDialog(this);
                        if (returnVal == JFileChooser.APPROVE_OPTION){
                                File file = chooser.getSelectedFile();
                                setTitle(file.getAbsolutePath());
                                fileAdress = file.getAbsolutePath();
                        }
        }

        private void openFasta() {
                        chooser.setFileFilter(new FastaFilter());
                        int returnVal = chooser.showOpenDialog(this);
                        try {
                                if (returnVal == JFileChooser.APPROVE_OPTION){
                                        File file = chooser.getSelectedFile();
                                FileInputStream input = new FileInputStream(file);
                                        fastaList = (ArrayList<FastaSequence>) SequenceUtil.readFasta(input);
                                        setTitle(file.getAbsolutePath());
                                        System.out.println("配列数"+fastaList.size());
                                }
                        } catch (FileNotFoundException event) {
                                event.printStackTrace();
                        } catch (IOException event) {
                                event.printStackTrace();
                        }
        }

        //ファイルを保存するメソッド
        private void saveFile() {
                        chooser.setFileFilter(new TxtFilter());
                        int returnVal = chooser.showSaveDialog(this);
                        try {
                                if (returnVal == JFileChooser.APPROVE_OPTION) {
                                        FileWriter fWriter = new FileWriter(chooser.getSelectedFile());
                                        textArea.write(fWriter);
                                }
                        } catch (FileNotFoundException event) {
                                event.printStackTrace();
                        } catch (IOException event) {
                                event.printStackTrace();
                        }
        }
        //run メソッド
        private void runOutFile() throws IOException {
                        FileReader fReader;
                        fReader = new FileReader(fileAdress);
                        BufferedReader bReader = new BufferedReader(fReader);

                        int threshold = 60;//ヒットした領域の切る値（%）
                        String line = "start";
                        int blastx = 0;
```

```
String ppId = null;
boolean indelcheck = false;
int hitCount = 0;
seqlist = new ArrayList();
indelList = new ArrayList();
while(line.indexOf("Number of sequences in database:") == -1) {
        line = bReader.readLine();
        if(line.indexOf("BLASTX")!=-1){
                blastx = 1;
        }

        // 1 つのクエリーについて読み始める
        if(line.indexOf("Query=")!=-1){
                String[] id = line.split("¥¥s");
    String name = id[1];
                float querylength =0;

        if (line.indexOf("length=")!=-1){
                String[] length = line.split("¥¥s¥¥s");
                querylength = Integer.parseInt(length[1].replaceAll("[^0-9]",""),10);
                if (blastx == 1) {
                        querylength = querylength/3;
                }
        }

        // 1 つの遺伝子内について、情報を集める
        int hit = -1;
        String geneName = null;
                String geneId = null;
                hitareaList = new ArrayList();
                float hitlength = 0;
                float identities = 0;
                int geneNo = 0;//ヒットした遺伝子の最初のもののみを読むためのパラメータ

        while ( line.indexOf("Effective search space used:") == -1 ){
                line = bReader.readLine();
                if (line.indexOf("Length=")!=-1){
                        String[] length = line.split("=");
                querylength = Integer.parseInt(length[1].replaceAll("[^0-9]",""),10);
                if (blastx == 1) {
                        querylength = querylength/3;
                }
                }
                //ブラストヒットした場合の処理
                else if(line.indexOf("Sequences producing significant alignments") != -1){
                        //hit = 1;
                        hitCount++;
                        while ( line.indexOf("Effective search space used:") == -1 ){
                                line = bReader.readLine();
                                if (line.indexOf(">") == 0){
                                        geneNo++;
                                }
                                //ブラストヒットの一番最初の遺伝子についての
読み取り

                                if (geneNo == 1){
                                        if(line.indexOf("name")!=-1){
                                                //name                に
geneName を代入して次のベクトルへ
                                                String  regex  =
"name=.+?;";
                                                Pattern     p     =
Pattern.compile(regex);
                                                Matcher     m     =
p.matcher(line);

                                                if(m.find()){

                int start = m.start()+5;

                int end = m.end()-1;
                                                                if
(geneName == null){

                geneName = line.substring(start,end);
                                                                }
                                                                if
(line.indexOf("parent=") != -1) {

                String regex2 = "FBgn.....";

                Pattern p2 = Pattern.compile(regex2);

                Matcher m2 = p2.matcher(line);


                if(m2.find()){

                        if (geneId == null){
```

```
                    geneId = m2.group();
                }else if(geneId == m2.group()) {

                }else {
                    System.out.println("複数ヒットです");
                }
            }

            System.out.println("name マッチしません");
-1){
代入して次のベクトルへ
"FBgn.....";
Pattern.compile(regex);
p.matcher(line);

(geneId == null){
            geneId = m.group();
            }else if(geneId == m.group()) {

            }else {
            System.out.println("複数ヒットです");


=") != -1){
bReader.readLine();
イトの位置

0;
0;
0://最初にヒットした遺伝子内の個所の、先頭の部分のクエリー番号のみを記録する
= false;
の読み取り
== false){
            if(line.indexOf("Identities =") != -1){
            String[] ident = line.split("¥¥s");
            String length[] = ident[3].split("/");
            identScore = Integer.parseInt(ident[4].replaceAll("[^0-9]",""),10);
            localLength = Integer.parseInt(length[1].replaceAll("[^0-9]",""),10);

            if((line.indexOf("Frame =")!=-1) && (blastx == 1)){
            String[] fr = line.split(" ");
            int last = fr.length - 1;
            frame = Integer.parseInt(fr[last]);
                                                    }
                                                }
                                                else {

                                                }
                                            }
                                            else if(line.indexOf("FBgn") !=
                                                //ID に geneId に
                                                String regex =
                                                Pattern p =
                                                Matcher m =

                                                if(m.find()){
                                                    if

                                                    }
                                                }
                                            }
                                            else if (line.indexOf("Score
                                                line =
                                                int frame = 0;
                                                int start = 0://サ
                                                int end = 0;
                                                int identScore =
                                                int localLength =
                                                int siteCount =
                                                boolean lastarea
                                                //次のエリアまで
                                                while (lastarea

                                                }
```

```
}
if
((line.indexOf("Query") != -1) && (identScore > threshold)){
        String[] siteline = line.split("    ",0);
        if (siteCount == 0){
                start = Integer.parseInt(siteline[1].replaceAll("[^0-9]",""),10);
        }
        end = Integer.parseInt(siteline[siteline.length-1].replaceAll("[^0-9]",""),10);
        siteCount++;
}
//
ヒット 2 箇所目以降
        else if ((identScore > threshold) && (line.indexOf("Score =") != -1)){
        hitareaList.add(new IndelData(start,end,frame,identScore,localLength));
        hit = 1;
        siteCount = 0;
}
        //1 つ目の遺伝子が終わり、次の遺伝子か 1 つの遺伝子のみについての情報を読み取ったら終了
        else if ((line.indexOf(">") == 0 ) | | (line.indexOf("Lambda") != -1)) {
        geneNo++;
        lastarea = true;
        if (identScore > threshold){
                hit = 1;
                hitareaList.add(new IndelData(start,end,frame,identScore,localLength));
        }
        //ヒットした部分の長さの和を計算。変則ヒットは計算していない

        if (hitareaList.size() > 1){
                if (hitareaList.size() > 4){
                        System.out.println(name + ":"+hitareaList.size()+"個所を結合");
                        ArrayList fix = new ArrayList();
                        for (int j=0;j<5;j++){
                                fix.add(hitareaList.get(j));
                        }
                        hitareaList = fix;
                        System.out.println("ヒット数大杉");
                }
                Collections.sort(hitareaList,new IndelComparator());
                int pHitlength=0;
                for (int j=0;j<hitareaList.size();j++){
                        pHitlength += hitareaList.get(j).getLength();
                }
                for (int j=0;j<hitareaList.size()-1;j++){
                        int reverse = hitareaList.get(j).getFrame();
                        int preS = hitareaList.get(j).getStartSite();
                        int preE = hitareaList.get(j).getEndSite();
                        int proS = hitareaList.get(j+1).getStartSite();
```

```
                                int proE = hitareaList.get(j+1).getEndSite();
                                //System.out.println("前のシーケンス:"+preS+","+preE+"後のシーケンス:"+proS+","+proE);
                                int dif = 0;
                                if (reverse < 0){
                                        if (preE < proS){
                                                dif = proS - preE;
                                        }
                                }
                                else if (reverse > 0) {
                                        if (preE > proS ){
                                                dif = preE - proS;
                                        }
                                }
                                if (blastx == 1){
                                        dif /= 3;
                                }
                                pHitlength -= dif;
                        }
                        hitlength = pHitlength;
                }else if (hitareaList.size() == 1){
                        hitlength = hitareaList.get(0).getLength();
                }else if (hitareaList.size() == 0){
                        hit = -1;
                        //System.out.println("hit したが、threshold でアウト");
                }
                break;
                                                                                                                                        }
                line = bReader.readLine();
                                                                                                                                }
                                                                                                                        }
                                                                                                                }
                                                                                                        }
                                                                                                }
                                                                                        }
                                                                                }
                                                                                //ブラストヒットしなかった場合
                                                                                else if(line.indexOf("No hits found")!=-1){
                                                                                        //indentities に 0 を代入して次のベクトルへ
                                                                                        identities = 0;
                                                                                }
                                                        }
                                                        //遺伝子内情報を集めた後、seqData の生成
                                                        if(line.indexOf("Effective search space used:") != -1){
                                                                //シーケンスリストの生成
                                                                identities = (hitlength/querylength)*100;
                                                                seqlist.add(new SeqData(name,hit,identities,geneName,geneId,geneNo));
                                                                indelList.add(hitareaList);
                                                        }
                                                }
                                        }
                                        bReader.close();
                                        fReader.close();
                                        System.out.println("sequenceList:"+seqlist.size());
                                        System.out.println("HitCount:"+hitCount);
                                        System.out.println("IndelList:" + indelList.size());
                }

                private void removeHit() throws FileNotFoundException {
                        int hitcount = 0;
                        for(int i=0;i<seqlist.size();i++) {
                                int hit = ((SeqData) seqlist.get(i)).getHit();
                                if (hit == 1) {
```

```
                                        hitcount++;
                                        fastaList.remove(i-hitcount+1);
                                        System.out.println(i);
                                }else if (hit == -1) {
                                        System.out.println("no hit");
                                }

                        }

                        chooser.setFileFilter(new FastaFilter());
                        int returnVal = chooser.showSaveDialog(this);
                        try {
                                if (returnVal == JFileChooser.APPROVE_OPTION) {
                                        OutputStream os = new FileOutputStream(chooser.getSelectedFile());
                                    SequenceUtil.writeFasta(os, fastaList);
                                }
                        } catch (FileNotFoundException event) {
                                event.printStackTrace();
                        } catch (IOException event) {
                                event.printStackTrace();
                        }
                }

                private void saveHit() throws FileNotFoundException {
                        textArea.append("変則ヒットリスト¥n");
                        for(int i=0;i<seqlist.size();i++) {//out ファイルのリスト
                                int hit = ((SeqData) seqlist.get(i)).getHit();
                                int paralog = ((SeqData) seqlist.get(i)).getGeneNo();
                                String flybaseId    = ((SeqData) seqlist.get(i)).getId();
                                if (hit == -1) {
                                        //System.out.println(((SeqData) seqlist.get(i)).getName()+":ヒットなし");
                                }
                                else if (hit == 1) {
                                        System.out.println(fastaList.get(i).getId() + " Hit:" + ((SeqData) seqlist.get(i)).getHit());
                                        String seq = fastaList.get(i).getSequence();//fasta のリスト
                                        ArrayList<IndelData> indellist = indelList.get(i);

                                        seq = modseq(seq,indellist,false);
                                        if (indellist.size() > 4){
                                                textArea.append("ヒット数が多すぎます:" + flybaseId);
                                        }

                                        String id = ((SeqData) seqlist.get(i)).getGene()+ "(" +((SeqData) seqlist.get(i)).getId() + ")
localName:" + ((SeqData) seqlist.get(i)).getName() + "frame:" + indellist.get(0).getFrame();
                                        FastaSequence fs = new FastaSequence(id,seq);
                                        fastaList2.add(fs);
                                }
                        }
                        chooser.setFileFilter(new FastaFilter());
                        int returnVal = chooser.showSaveDialog(this);
                        try {
                                if (returnVal == JFileChooser.APPROVE_OPTION) {
                                        OutputStream os = new FileOutputStream(chooser.getSelectedFile());
                                    SequenceUtil.writeFasta(os, fastaList2);
                                }
                        } catch (FileNotFoundException event) {
                                event.printStackTrace();
                        } catch (IOException event) {
                                event.printStackTrace();
                        }
                }
                //indel リストを並べ替え、入れ子になっているものを修正
                private ArrayList<IndelData> fixindel(ArrayList<IndelData> indelList,boolean onlyHitArea){
                        Collections.sort(indelList,new IndelComparator());
                        ArrayList<IndelData> fixlist = new ArrayList();
                        fixlist.add(indelList.get(0));
                        for(int i=0;i<indelList.size()-1;i++){
                                int reverse = indelList.get(i).getFrame();
                                int preS = indelList.get(i).getStartSite();
                                int preE = indelList.get(i).getEndSite();
                                int proS = indelList.get(i+1).getStartSite();
                                int proE = indelList.get(i+1).getEndSite();
                                int preIdent = indelList.get(i).getIdent();
                                int proIdent = indelList.get(i+1).getIdent();
                                int same = 0;

                                if (onlyHitArea){
                                        if (reverse > 0){
                                                if (preS < proS){
                                                        if (preE > proE){
                                                                fixlist.add(new
IndelData(preS,proS-1,indelList.get(i).getFrame(),0,proS -1 - preS));
                                                                fixlist.add(indelList.get(i+1));
                                                                fixlist.add(new
IndelData(proE+1,proS-1,indelList.get(i).getFrame(),0,proS -1 - preS));
                                                        }else{
                                                                if(preIdent > proIdent){
                                                                        fixlist.add(indelList.get(i));
                                                                        fixlist.add(new
IndelData(preE+1,proE,indelList.get(i).getFrame(),0,proE-preE));
```

```
							}else{
								fixlist.add(new
IndelData(preS,proS-1,indelList.get(i).getFrame(),0,proS-1-preS));
								fixlist.add(indelList.get(i+1));
							}
						}

					}else if (preS == proS){
						if(preIdent > proIdent){
							fixlist.add(indelList.get(i));
							fixlist.add(new
IndelData(preE+1,proE,indelList.get(i).getFrame(),0,proE-preE));
						}else{
							fixlist.add(new
IndelData(preS,proS-1,indelList.get(i).getFrame(),0,proS-1-preS));
							fixlist.add(indelList.get(i+1));
						}
					}
				}else{

				}
			}
			if ((preS == proS) && (preE == proE)){
				if ((same == 0) && (i > 0)){
					fixlist.add(indelList.get(i));
				}
				same++;
			}
			if(reverse > 0){
				if(proE <= preE){
					fixlist.remove(fixlist.size() - 1);
					if ((preS == proS) | | (preE == proE)){
						fixlist.add(new
IndelData(preS,preE,indelList.get(i).getFrame(),0,preS-preE));

					}else{
						fixlist.add(new
IndelData(preS,proS,indelList.get(i).getFrame(),0,proS-preS));
						fixlist.add(indelList.get(i+1));
						fixlist.add(new
IndelData(proE,preE,indelList.get(i).getFrame(),0,preE-proE));
						i++;
					}
				}else{
					fixlist.add(indelList.get(i+1));
				}
			}else{
				if(proE >= preE){
					fixlist.remove(fixlist.size() - 1);
					if ((preS == proS) | | (preE == proE)){
						fixlist.add(new
IndelData(preS,preE,indelList.get(i).getFrame(),0,preE-preS));
					}else{
						fixlist.add(new
IndelData(preS,proS,indelList.get(i).getFrame(),0,preS-proS));
						fixlist.add(indelList.get(i+1));
						fixlist.add(new
IndelData(proE,preE,indelList.get(i).getFrame(),0,proE-preE));
						i++;
					}
				}else{
					fixlist.add(indelList.get(i+1));
				}
			}

		}
		Collections.sort(indelList,new IndelComparator());
		return fixlist;
	}
	private String modseq(String initSeq,ArrayList<IndelData> indellist, boolean onlyHitArea){
		String seq = initSeq;
		int seqlength = seq.length();
		//System.out.println("initLength:" + seqlength);
		indellist = fixindel(indellist,false);
		int multihit = indellist.size();

		//ヒット一か所目の修正
		IndelData first = indellist.get(0);
		int frame = first.getFrame();
		int frameShift = 0;
		int presentSite = first.getStartSite();
		boolean hensokuhit = false;
		//表裏をもとに戻す
		if (frame < 0) {
			DNASequence ds = new DNASequence(seq);
			seq = ds.getReverseComplement().getSequenceAsString();
			frame = frame * (-1);
		}
		/*
		if (onlyHitArea) {
			String modseq = "";
```

```
                        for (int i=0;i<multihit;i++){
                                frame = indellist.get(i).getFrame();
                                int ss = indellist.get(i).getStartSite() - 1;
                                int es = indellist.get(i).getEndSite() - 1;
                                if (frame < 0){
                                        ss = seqlength - ss - 1;
                                        es = seqlength - es - 1;
                                }
                                modseq += seq.substring(ss,es+1);
                        }
                        seq = modseq;
                }else{

                }*/
                StringBuilder sb = new StringBuilder(seq);
                if (onlyHitArea == false){
                        switch (frame){
                        case 2:seq = sb.delete(0, 1).toString();frameShift = 2;break;
                        case 3:seq = sb.delete(0, 2).toString();frameShift++;break;
                        default:break;
                        }
                }
                int firstShift = frameShift;

                //2 個所目以降のサイトの位置がずれる（フォワードの場合）
                //2 個所目以降の修正
                if (multihit > 1){
                        if (multihit > 4) {

                        }else{
                                for (int j=0;j<multihit-1;j++){
                                        IndelData preindel = (IndelData) indellist.get(j);
                                        IndelData proindel = (IndelData) indellist.get(j+1);
                                        int preframe = preindel.getFrame();
                                        int proframe = proindel.getFrame();
                                        sb = new StringBuilder(seq);
                                        presentSite = preindel.getEndSite() + 1 - frameShift;
                                        if (proindel.getFrame() < 0){
                                                presentSite = seqlength - (preindel.getEndSite() + 1) ;
                                                preframe = preframe*(-1);
                                                proframe = proframe*(-1);
                                        }
                                        switch (preframe){
                                        case 2:
                                                switch (proframe){
                                                case 3:seq =
sb.delete(presentSite,presentSite+1).toString();frameShift++;break;
                                                case 1:seq =
sb.delete(presentSite,presentSite+2).toString();frameShift = frameShift + 2;break;
                                                default:break;
                                                }
                                                break;
                                        case 3:
                                                switch (proframe){
                                                case 1:seq =
sb.delete(presentSite,presentSite+1).toString();frameShift++;break;
                                                case 2:seq =
sb.delete(presentSite,presentSite+2).toString();frameShift = frameShift + 2;break;
                                                default:break;
                                                }
                                                break;
                                        case 1:
                                                switch (proframe){
                                                case 2:seq =
sb.delete(presentSite,presentSite+1).toString();frameShift++;break;
                                                case 3:seq =
sb.delete(presentSite,presentSite+2).toString();frameShift = frameShift + 2;break;
                                                default:break;
                                                }
                                                break;
                                        }
                                }
                        }

                }

                if (onlyHitArea && multihit < 5){
                        //ヒット領域のみを抽出
                        frame = first.getFrame();
                        int firstFrame = frame;
                        if (firstFrame < 0){
                                firstFrame = firstFrame*(-1);
                        }
                        switch (firstFrame){
                        case 1:firstFrame = 0;break;
                        case 2:firstFrame = 2;break;
                        case 3:firstFrame = 1;break;
                        }
                        int ss = first.getStartSite() - 1;
                        int es = indellist.get(indellist.size() - 1).getEndSite() - 1;
```

```
                        if (frame > 0){
                                es = es - frameShift;
                        }else if (frame < 0){
                                ss = seqlength - ss - 1;
                                es = seqlength - frameShift - es - 1;
                        }
                        seq = seq.substring(ss,es+1);
                }

                //seq = seq.replaceAll("N","");
                return seq;
        }
}
```

Class of information of insertion and deletion sites

```
package practice;

public class IndelData {
		private int startSite;
		private int frame;
		private int endSite;
		private int identity;
		private int length;

		public IndelData(int startSite,int endSite,int frame,int identity,int length) {
				this.startSite = startSite;
				this.endSite = endSite;
				this.frame = frame;
				this.identity = identity;
				this.length = length;
		}

		public int getStartSite(){
				return startSite;
		}
		public int getEndSite(){
				return endSite;
		}
		public int getFrame(){
				return frame;
		}
		public int getIdent(){
				return identity;
		}
		public int getLength(){
				return length;
		}

}
```

Class of sequence data information of Blast

```
package practice;

public class SeqData {
		private String name;
		private int hit;
		private float identities;
		private String geneName;
		private String geneId;
		//private int frame;
		private int geneNo;

		public SeqData(String name,int hit,float identities,String geneName,String geneId,int geneNo) {
				this.name = name;
				this.hit = hit;
				this.identities = identities;
				this.geneName = geneName;
				this.geneId = geneId;
				//this.frame = frame;
				this.geneNo = geneNo;
		}
		public String getName() {
				return name;
		}
		public int getHit(){
				return hit;
		}
		public float getIdentities(){
				return identities;
		}
		public String getGene(){
				return geneName;
		}
		public String getId(){
				return geneId;
		}
		/*public int getFrame(){
				return frame;
		}*/
		public int getGeneNo(){
				return geneNo;
		}

}
```

Indel comparator

```
package practice;

import java.util.Comparator;

public class IndelComparator implements Comparator<IndelData> {

		public IndelComparator() {
			// TODO Auto-generated constructor stub
		}
		@Override
		public int compare(IndelData o1, IndelData o2) {
			int frame1 = o1.getFrame();
			int s1 = o1.getStartSite();
			int s2 = o2.getStartSite();
			if (frame1 < 0){
				if (s1 > s2){
					return -1;
				}
				else{
					return 1;
				}
			}else if (frame1 > 0){
				if (s1 > s2){
					return 1;
				}
				else{
					return -1;
				}
			}else{
				return 0;
			}
		}

}
```

Flybase gene number filter

```
package practice;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class FbgnFilter {

	public FbgnFilter() {
		// TODO Auto-generated constructor stub
	}
	public static String fbgnfilter(String line){
		String regex2 = "FBgn.....";
		Pattern p2 = Pattern.compile(regex2);
		Matcher m2 = p2.matcher(line);

		if(m2.find()){
			return m2.group();
		}
		else {
			return "error";
		}
	}
	public static String contigfilter(String line){
		String regex = "contig.....";
		Pattern p = Pattern.compile(regex);
		Matcher m = p.matcher(line);

		if(m.find()){
			return m.group();
		}
		else {
			return "error";
		}
	}
	public static String parenthesisfilter(String line){
		String regex = "¥¥(.+¥¥)";
		Pattern p = Pattern.compile(regex);
		Matcher m = p.matcher(line);

		if(m.find()){
			return m.group();
		}
		else {
			return "error";
		}
	}

}
```

Blast output file filter

```
package practice;
import java.io.File;
import javax.swing.filechooser.FileFilter;

public class OutFilter extends FileFilter {

	public boolean accept (File file) {
		if (file !=null) {
			//ディレクトリ判定
			if (file.isDirectory()) {
				return true;
			}else {
				//拡張子判定
				String ext = getExtension(file);
				if (ext != null && ext.equals("out")) {
					return true;
				}
			}

		}
		//true が返らなかったとき
		return false;
	}

	//拡張子を取得するメソッド
	private String getExtension(File file) {
		if(file == null) {
			return null;
		} else {
			//ファイル名を取得
			String name = file.getName();
			//最後のピリオド位置を取得
			int period = name.lastIndexOf('.');
			if (period > 0 && period < name.length() -1 ) {
				//拡張子を小文字で返す
				return name.substring(period + 1).toLowerCase();
			} else {
				return null;
			}
		}
	}

	public String getDescription() {
		return"Out ファイル(*.out)";
	}

}
```

Orthologous gene checker by Flybase Gene numbers

```
package practice;
import javax.swing.*;
import javax.swing.text.DefaultEditorKit.*;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;
import org.biojava3.core.sequence.io.*;
import org.biojava3.core.sequence.*;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


@SuppressWarnings("serial")
public class OrthologChecker extends JFrame implements ActionListener{

		private JTextArea textArea = new JTextArea();

		//種名入力欄
		private JLabel splabel = new JLabel("種名");
		private JTextField species = new JTextField(10);

		//プログラム実行部分
		private JButton openFasta = new JButton("fasta を開く");
		private JButton save = new JButton("オーソログリストをテキストで保存");
		private JButton setFolder = new JButton("fasta の保存先");
		private JButton check = new JButton ("checkOrtholog");
		private JFileChooser chooser;
		private List<FastaSequence> fastaList;
		/*
		private ArrayList<FastaSequence> fastaListOutNuc = new ArrayList<FastaSequence>();
		private ArrayList<FastaSequence> fastaListOutPro = new ArrayList<FastaSequence>();
		*/
		private ArrayList<FastaSequence> fastaList3 = new ArrayList<FastaSequence>();//最後に、NGS データのみをまとめたリスト
		String saveadress = "C:¥¥cDNAData";

		//コンストラクタ
		public OrthologChecker(){
			super("Ortholog 結果解析");
			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			setSize(800,600);


			JPanel panel1 = new JPanel();
			JPanel panel2 = new JPanel();
			getContentPane().add(panel1,BorderLayout.NORTH);
			getContentPane().add(new JScrollPane(textArea),BorderLayout.CENTER);
			getContentPane().add(panel2,BorderLayout.SOUTH);

			panel1.add(openFasta);
			panel1.add(check);
			panel1.add(save);
			panel2.add(splabel);
			panel2.add(species);
			panel2.add(setFolder);

			openFasta.addActionListener(this);
			save.addActionListener(this);
			check.addActionListener(this);
			setFolder.addActionListener(this);

		}

		public static void main(String[] args) {
			new OrthologChecker().setVisible(true);

		}

		//[開く][保存]ボタンが押された時の処理
		public void actionPerformed(ActionEvent event){
			if (event.getSource().equals(openFasta)) {
				openFasta();
			}
			else if (event.getSource().equals(setFolder)){
				try {
					setFolder();
				} catch (FileNotFoundException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				} catch (IOException e) {
					// TODO Auto-generated catch block
```

```
                                        e.printStackTrace();
                                }
                }
                else if (event.getSource().equals(save)){
                                save();
                }
                else if (event.getSource().equals(check)){
                                try {
                                        check();
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                }

}

//ファイルを開くメソッド
private void openFasta() {
                chooser = new JFileChooser("C:¥¥cDNAData");
                chooser.setFileFilter(new FastaFilter());
                int returnVal = chooser.showOpenDialog(this);
                try {
                                if (returnVal == JFileChooser.APPROVE_OPTION){
                                        File file = chooser.getSelectedFile();
                        FileInputStream input = new FileInputStream(file);
                                        fastaList = SequenceUtil.readFasta(input);
                                        setTitle(file.getAbsolutePath());

                                        System.out.println("配列数"+fastaList.size());
                                        System.out.println(((FastaSequence)fastaList.get(3)).getId());

                                }
                } catch (FileNotFoundException event) {
                                event.printStackTrace();
                } catch (IOException event) {
                                event.printStackTrace();
                }
}

//ファイルを保存するメソッド
private void save() {
                int returnVal = chooser.showSaveDialog(this);
                try {
                                if (returnVal == JFileChooser.APPROVE_OPTION) {
                                        FileWriter fWriter = new FileWriter(chooser.getSelectedFile());
                                        textArea.write(fWriter);

                                }
                } catch (FileNotFoundException event) {
                                event.printStackTrace();
                } catch (IOException event) {
                                event.printStackTrace();
                }
}
//setFolder メソッド
private void setFolder() throws FileNotFoundException,IOException {
                chooser = new JFileChooser("C:¥¥cDNAData");
                chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                int returnVal = chooser.showOpenDialog(this);
                if (returnVal == JFileChooser.APPROVE_OPTION){
                                File file = chooser.getSelectedFile();
                                setTitle(file.getAbsolutePath());
                                saveadress = file.getAbsolutePath();
                    }
                }
//check メソッド
 private void check() throws FileNotFoundException,IOException {
                File file = new File("C:¥¥cDNAData¥¥originalAlignments");
                String[] orthologlist = file.list(new FastaNameFilter());
                setTitle(file.getAbsolutePath());
                //String fileAdress = file.getAbsolutePath();
    String spname = species.getText();
    int count = 0;
    int multicount = 0;
    List<FastaSequence> notOrtholog = new ArrayList();
    boolean hit;

    for (int i=0; i< fastaList.size(); i++){
                String name = fastaList.get(i).getId();
                String id = FbgnFilter.fbgnfilter(name);
                hit = false;
                for (int j=0; j< orthologlist.length; j++) {
                                        if (orthologlist[j].indexOf(id) != -1) {
                                                hit = true;
                                                String seq = fastaList.get(i).getSequence();
                                                //contig 名の読み取り
                                                String contig = FbgnFilter.contigfilter(name);
                                                //ファイルへの書き込み設定
                                                String nucName = saveadress+"¥¥" + id + ".fasta";
```

```
                                                String proName = saveadress+"¥¥Prot¥¥" + id +".fasta";


                                                //ヌクレオチドシーケンスの作成
                                                FastaSequence nucFs = new FastaSequence(spname + "(" + contig + ")",seq);
                                                ArrayList<FastaSequence> nucList = new ArrayList<FastaSequence>();

                                                //既存ファイルの確認
                                                File existNuc = new File(nucName);
                                                if (existNuc.exists()) {
                                                        nucName = saveadress + "¥¥" + id + "(" + contig + ")"+".fasta";
                                                }
                                                nucList.add(nucFs);
                                                FastaSequence fsLast = new FastaSequence(id + "(" + contig + ")",seq);

                                                //アミノ酸シーケンスの作成
                                                DNASequence ds = new DNASequence(seq);
                                                RNASequence rs = ds.getRNASequence();
                                                ProteinSequence ps = rs.getProteinSequence();
                                                String psString = ps.getSequenceAsString();
                                                FastaSequence proFs = new FastaSequence(spname + "(" + contig +
")",psString);

                                                ArrayList<FastaSequence> proList = new ArrayList<FastaSequence>();
                                                //既存ファイルの確認
                                                File existPro = new File(proName);
                                                if (existNuc.exists()) {
                                                        proName = saveadress+"¥¥Prot¥¥" + id +"(" + contig +
")"+".fasta";
                                                }
                                                proList.add(proFs);

                                                //オーソログ遺伝子の表示、リストのストック
                                                textArea.append(id+"¥n");
                                                fastaList3.add(fsLast);
                                                if (proList.size()>1){
                                                        System.out.println(id+"複数ヒットしました。");
                                                        multicount++;
                                                }
                                                count++;

                                                OutputStream nucOs = new FileOutputStream(nucName);
                                                SequenceUtil.writeFasta(nucOs, nucList);
                                                nucOs.close();
                                                OutputStream proOs = new FileOutputStream(proName);
                                                SequenceUtil.writeFasta(proOs, proList);
                                                proOs.close();
                                        }
                                }
                        if (hit == false){
                                notOrtholog.add(fastaList.get(i));
                        }

                }
                String otherGeneAdress = saveadress + "¥¥notOrthologList.fasta";
                OutputStream notOrthologOs = new FileOutputStream(otherGeneAdress);
                        SequenceUtil.writeFasta(notOrthologOs, notOrtholog);
                        notOrthologOs.close();
                textArea.append("全"+ count + "遺伝子座でオーソログが見つかりました。");
                System.out.println("全"+ multicount + "遺伝子座で複数ヒットしました");
                int returnValue = chooser.showSaveDialog(this);
                        try {
                                if (returnValue == JFileChooser.APPROVE_OPTION) {
                                        OutputStream os = new FileOutputStream(chooser.getSelectedFile());
                                    org.biojava3.data.sequence.SequenceUtil.writeFasta(os, fastaList3);


                                }
                        } catch (FileNotFoundException event) {
                                event.printStackTrace();
                        } catch (IOException event) {
                                event.printStackTrace();
                        }
        }

}
```

Fasta name filter

```
package practice;
import java.io.File;
import java.io.FilenameFilter;

public class FastaNameFilter implements FilenameFilter {

	public FastaNameFilter() {
		// TODO Auto-generated constructor stub
	}

	@Override
	public boolean accept(File dir, String name) {
		int index = name.lastIndexOf(".");//拡張子の"."を探す
		//"."以下の文字列を取り出して全て小文字に
		String ext = name.substring(index+1).toLowerCase();
		//拡張子が"txt"と一致すれば取り出す
		if(ext.indexOf("fas")!= -1) {
			return true;
		}
		//それ以外のファイルはリストアップしない
		return false;
	}

}
```

Fasta filter

```
package practice;
import java.io.File;
import javax.swing.filechooser.FileFilter;

public class FastaFilter extends FileFilter {

	public boolean accept (File file) {
		if (file !=null) {
			//ディレクトリ判定
			if (file.isDirectory()) {
				return true;
			}else {
				//拡張子判定
				String ext = getExtension(file);
				if (ext != null && ext.indexOf("fas") != -1) {
					return true;
				}
			}

		}
		//true が返らなかったとき
		return false;
	}

	//拡張子を取得するメソッド
	private String getExtension(File file) {
		if(file == null) {
			return null;
		} else {
			//ファイル名を取得
			String name = file.getName();
			//最後のピリオド位置を取得
			int period = name.lastIndexOf('.');
			if (period > 0 && period < name.length() -1 ) {
				//拡張子を小文字で返す
				return name.substring(period + 1).toLowerCase();
			} else {
				return null;
			}
		}
	}

	public String getDescription() {
		return"Fasta  ファイル(*.fasta)";
	}

}
```

```
package practice;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JButton;
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.swing.Action;

import org.biojava3.data.sequence.FastaSequence;
import javax.swing.JTextArea;
import java.awt.Scrollbar;
import javax.swing.JScrollPane;

public class ProfileAlignmentbyClustal extends JFrame {

	private JPanel contentPane;
	private JTextField newSeqAdress;
	private JTextField orthologAdress;
	private JTextField textField_2;
	private JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
	private final Action action = new SwingAction();
	private final Action action_1 = new SwingAction_1();
	private final Action action_2 = new SwingAction_2();
	private JTextArea textArea = new JTextArea();
	private ArrayList seqlist;
	private ArrayList indelList;//ヒットエリアリストのリスト、seqlist と同じ数だけ存在
	private ArrayList hitareaList;//各配列内のインデルリスト
	private ArrayList<FastaSequence> fastaList = new ArrayList<FastaSequence>();
	private ArrayList<FastaSequence> fastaList2 = new ArrayList<FastaSequence>();
	private final Action action_3 = new SwingAction_3();
	private final Action action_4 = new SwingAction_4();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					ProfileAlignmentbyClustal frame = new ProfileAlignmentbyClustal();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public ProfileAlignmentbyClustal() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 571, 633);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		newSeqAdress = new JTextField();
		newSeqAdress.setColumns(10);

		JLabel label = new
JLabel("¥u65B0¥u305F¥u306B¥u30A2¥u30E9¥u30A4¥u30E1¥u30F3¥u30C8¥u3059¥u308B¥u7A2E¥u3092¥u542B¥u3080¥u30D5¥u30A9¥u30EB
¥u30C0");

		JButton btnNewButton = new JButton("¥u53C2¥u7167");
		btnNewButton.setAction(action);
```

```
                JButton button = new JButton("¥u53C2¥u7167");
                button.setAction(action_1);

                orthologAdress = new JTextField();
                orthologAdress.setText("C:¥¥cDNAData¥¥originalAlignments¥¥Prot¥¥");
                orthologAdress.setColumns(10);

                JLabel label_1 = new JLabel("¥u30AA¥u30FC¥u30BD¥u30ED¥u30B0¥u30D5¥u30A9¥u30EB¥u30C0");

                JLabel label_2 = new JLabel("¥u30D7¥u30ED¥u30D5¥u30A1¥u30A4¥u30EB¥u30A2¥u30E9¥u30A4¥u30E1¥u30F3¥u30C8¥u6E08¥u307F¥u306E¥u7A2E¥u3092¥u542B¥u3080¥u30D5¥u30A9¥u30EB¥u30C0");

                textField_2 = new JTextField();
                textField_2.setColumns(10);

                JButton button_1 = new JButton("¥u53C2¥u7167");
                button_1.setAction(action_2);

                JButton createBat = new JButton("¥u30D0¥u30C3¥u30C1¥u30D5¥u30A1¥u30A4¥u30EB¥u4F5C¥u6210");
                createBat.setAction(action_3);

                JScrollPane scrollPane = new JScrollPane();

                JButton btnMultiplealign = new JButton("multipleAlign");
                btnMultiplealign.setAction(action_4);
                GroupLayout gl_contentPane = new GroupLayout(contentPane);
                gl_contentPane.setHorizontalGroup(
                        gl_contentPane.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_contentPane.createSequentialGroup()
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addContainerGap()
                                                        .addComponent(scrollPane, GroupLayout.DEFAULT_SIZE, 467, Short.MAX_VALUE))
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addGap(12)
                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING, false)
                                                        .addComponent(label_2, Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(label_1, Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(label, Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, 244, Short.MAX_VALUE))
                .addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
                .addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING, false)
                .addComponent(textField_2, Alignment.LEADING)
                .addComponent(orthologAdress, Alignment.LEADING, GroupLayout.DEFAULT_SIZE, 159, Short.MAX_VALUE))
                                                        .addComponent(newSeqAdress, GroupLayout.PREFERRED_SIZE, 159, GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(ComponentPlacement.RELATED)
                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING, false)
                                                        .addComponent(btnNewButton, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(button_1, Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                        .addComponent(button, Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addContainerGap()
                                                        .addComponent(btnMultiplealign)
                                                        .addGap(96)
                                                        .addComponent(createBat)))
                                                .addGap(66))
                );
                gl_contentPane.setVerticalGroup(
                        gl_contentPane.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_contentPane.createSequentialGroup()
                                        .addGap(37)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                .addComponent(newSeqAdress, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addGap(3)
                                                        .addComponent(label))
                                                .addComponent(btnNewButton))
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addGap(10)
                                                        .addComponent(label_1))
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addGap(6)
```

```
                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)

                .addGroup(gl_contentPane.createSequentialGroup()
                                                                                                                .addGap(1)

                .addComponent(ortholog_adress, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                                                                                                .addComponent(button))))
                                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                                                                .addGap(10)
                                                                                                .addComponent(label_2))
                                                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                                                                .addGap(7)
                                                                                                .addComponent(textField_2,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                                                                .addGap(6)
                                                                                                .addComponent(button_1)))
                                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                                .addComponent(scrollPane,       GroupLayout.PREFERRED_SIZE,       403,
GroupLayout.PREFERRED_SIZE)
                                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                                                .addComponent(createBat)
                                                                                .addComponent(btnMultiplealign)))
                                );

                                scrollPane.setViewportView(textArea);
                                contentPane.setLayout(gl_contentPane);
                }
                private class SwingAction extends AbstractAction {
                                public SwingAction() {
                                                putValue(NAME, "参照");
                                                putValue(SHORT_DESCRIPTION, "Some short description");
                                }
                                public void actionPerformed(ActionEvent e) {
                                                filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                                                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                                                if (selected == JFileChooser.APPROVE_OPTION){
                                                                File file = filechooser.getSelectedFile();
                                                                newSeqAdress.setText(file.getAbsolutePath()); //ラベルの文字をファイル名に
                                                }

                                }
                }
                private class SwingAction_1 extends AbstractAction {
                                public SwingAction_1() {
                                                putValue(NAME, "参照");
                                                putValue(SHORT_DESCRIPTION, "Some short description");
                                }
                                public void actionPerformed(ActionEvent e) {
                                                filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                                                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                                                if (selected == JFileChooser.APPROVE_OPTION){
                                                                File file = filechooser.getSelectedFile();
                                                                orthologAdress.setText(file.getAbsolutePath()); //ラベルの文字をファイル名に
                                                }
                                }
                }
                private class SwingAction_2 extends AbstractAction {
                                public SwingAction_2() {
                                                putValue(NAME, "参照");
                                                putValue(SHORT_DESCRIPTION, "Some short description");
                                }
                                public void actionPerformed(ActionEvent e) {
                                                filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                                                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                                                if (selected == JFileChooser.APPROVE_OPTION){
                                                                File file = filechooser.getSelectedFile();
                                                                newSeqAdress.setText(file.getAbsolutePath()); //ラベルの文字をファイル名に
                                                }
                                }
                }
                private class SwingAction_3 extends AbstractAction {
                                public SwingAction_3() {
                                                putValue(NAME, "プロファイルアライメント用バッチファイル作成");
                                                putValue(SHORT_DESCRIPTION, "Some short description");
                                }
                                public void actionPerformed(ActionEvent e) {
                                                File file = new File(newSeqAdress.getText());
                                                String fileParentAdress = file.getParent();
                                                String ngslist[] = file.list();
                                    ArrayList al = new ArrayList();
                                    ArrayList scriptList = new ArrayList();
                                    ArrayList<String> doubledOrtholog = new ArrayList();

                                    for(int i=0;i<ngslist.length;i++){
                                                if (ngslist[i].indexOf(".fas")!= -1){
                                                                if ((ngslist[i].indexOf(".fasta")!= -1)) {
```

```
                                                al.add(ngslist[i]);
                                        }
                                                else {
                                                al.add(ngslist[i].replaceAll("fas","fasta"));
                                        }
                        }
                }
                for (int j=0;j<al.size();j++) {
                        String fn =(String)al.get(j);
                        String profile = " --is-profile";
                        String p1 = " --profile1="+fn;
                        String p2 = " --profile2="+orthologAdress.getText()+fn;
                        String out = " --outfile=aligned_"+fn;
                        String force = "";

                        /*
                        //前回のプロファイルアラインメントの有無をチェック
                        if (previousAlign){
                                for (int k=0;k<preOrthologlist.length;k++) {
                                        if (preOrthologlist[k].indexOf(fn)!=-1) {
                                                p2 = " --profile2="+preOrthologAdress+"aligned_"+fn;
                                                doubledOrtholog.add(preOrthologlist[k]);
                                        }
                                }
                        }*/

                        //重複するヒットの確認（カッコでくくられている文字列の有無をチェック）
                        String regex = "FBgn.....¥¥(" + ".*" +"¥¥)";
                        Pattern p = Pattern.compile(regex);
                                Matcher m = p.matcher(fn);
                        if (m.find()) {
                                fn = fn.substring(0,11)+".fasta";
                                p2 = " --profile2=aligned_"+fn;
                                out = " --outfile=aligned_"+fn;
                                profile = "";
                                force = " --force";
                        }

                        String clustal = "clustalo" + profile + p1 + p2 + out + force +"¥n";
                        scriptList.add(clustal);
                }

                        //順序の入れ替え、テキストエリアへの書き込み
                        Collections.reverse(scriptList);
                        for (int k=0;k<al.size();k++) {
                                textArea.append((String) scriptList.get(k));
                        }
                        /*
                        //重複したオーソログリストの生成
                        BufferedWriter writer;
                        try {
                                writer = new BufferedWriter(new FileWriter(newSeqAdress + "preOrthologList.txt"));
                                for (int l=0;l<doubledOrtholog.size();l++) {
                                        writer.write(doubledOrtholog.get(l));
                                writer.newLine();
                                }
                        writer.flush();
                        writer.close();
                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }*/
                }
}
private class SwingAction_4 extends AbstractAction {
                public SwingAction_4() {
                        putValue(NAME, "マルチプルアライメント用バッチファイルの作成");
                        putValue(SHORT_DESCRIPTION, "Some short description");
                }
                public void actionPerformed(ActionEvent e) {
                        File file = new File(newSeqAdress.getText());
                        String fileParentAdress = file.getParent();
                        String ngslist[] = file.list();
                ArrayList al = new ArrayList();
                ArrayList scriptList = new ArrayList();
                ArrayList<String> doubledOrtholog = new ArrayList();

                for(int i=0;i<ngslist.length;i++){
                        if (ngslist[i].indexOf(".fas")!= -1){
                                al.add(ngslist[i]);
                                /*
                                        if ((ngslist[i].indexOf(".fasta")!= -1)) {
                                        al.add(ngslist[i]);
                                }
                                        else {
                                        al.add(ngslist[i].replaceAll("fas","fasta"));
                                }*/
                        }
                }
                for (int j=0;j<al.size();j++) {
```

```
                                        String fn =(String)al.get(j);

                                        String in = " --infile="+fn;
                                        String out = " --outfile=MultipleAligned_"+fn;
                                        //String dist = " --distmat-out=" + "distmat" + fn.substring(0,11) + ".txt";
                                        String clustal = "clustalo" + in + out +"¥n";
                                        textArea.append(clustal);
                                }
                        }
                }
        }
```

Nucleotide and amino acid converter

```
package practice;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JFileChooser;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import javax.swing.Action;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import javax.swing.JCheckBox;

public class NucAminoConverter extends JFrame {

	private JPanel contentPane;
	private JTextField textField;
	private JTextField textField_1;
	private JButton btnAminofolder;
	private JTextArea textArea;
	private JCheckBox checkBox;
	private final Action action = new SwingAction();
	private final Action action_1 = new SwingAction_1();
	private final Action action_2 = new SwingAction_2();
	private final Action action_3 = new SwingAction_3();
	private final Action action_4 = new SwingAction_4();
	private final Action action_5 = new SwingAction_5();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					NucAminoConverter frame = new NucAminoConverter();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public NucAminoConverter() {
		setTitle("¥u30A2¥u30DF¥u30CE¥u9178¥u3001¥u30CC¥u30AF¥u30EC¥u30AA¥u30C1¥u30C9¥u5909¥u63DB");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 601, 476);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		textField = new JTextField();
		textField.setColumns(10);

		JButton btnBrowse = new JButton("browse");
		btnBrowse.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
			}
		});
		btnBrowse.setAction(action);

		textField_1 = new JTextField();
		textField_1.setColumns(10);

		btnAminofolder = new JButton("aminofolder");
		btnAminofolder.setAction(action_1);

		JScrollPane scrollPane = new JScrollPane();

		JButton btnNucconv = new JButton("nucConv");
		btnNucconv.setAction(action_2);
```

```
		JButton btnProconv = new JButton("proConv");
		btnProconv.setAction(action_3);

		JButton btnReciprocal = new JButton("reciprocal");
		btnReciprocal.setAction(action_4);

		checkBox = new JCheckBox("¥u30D7¥u30ED¥u30D5¥u30A1¥u30A4¥u30EB¥u30A2¥u30E9¥u30A4¥u30E1¥u30F3¥u30C8");
		checkBox.setSelected(true);

		JButton btnNewButton = new JButton("New button");
		btnNewButton.setAction(action_5);
		GroupLayout gl_contentPane = new GroupLayout(contentPane);
		gl_contentPane.setHorizontalGroup(
			gl_contentPane.createParallelGroup(Alignment.TRAILING)
				.addGroup(gl_contentPane.createSequentialGroup()
					.addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
		.addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
			.addComponent(scrollPane, GroupLayout.DEFAULT_SIZE, 563, Short.MAX_VALUE)
		.addGroup(gl_contentPane.createSequentialGroup()
		.addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
		.addComponent(textField_1, Alignment.LEADING, GroupLayout.DEFAULT_SIZE, 397, Short.MAX_VALUE)
		.addComponent(textField, GroupLayout.DEFAULT_SIZE, 397, Short.MAX_VALUE))
		.addPreferredGap(ComponentPlacement.RELATED)
		.addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
		.addComponent(btnAminofolder, GroupLayout.DEFAULT_SIZE, 159, Short.MAX_VALUE)
		.addComponent(btnBrowse, GroupLayout.DEFAULT_SIZE, 159, Short.MAX_VALUE)))
		.addGroup(gl_contentPane.createSequentialGroup()
			.addComponent(btnNucconv, GroupLayout.DEFAULT_SIZE, 122, Short.MAX_VALUE)
		.addPreferredGap(ComponentPlacement.RELATED)
			.addComponent(btnProconv, GroupLayout.DEFAULT_SIZE, 140, Short.MAX_VALUE)
		.addPreferredGap(ComponentPlacement.RELATED)
			.addComponent(btnReciprocal, GroupLayout.DEFAULT_SIZE, 124, Short.MAX_VALUE)
		.addPreferredGap(ComponentPlacement.RELATED)
			.addComponent(btnNewButton, GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
			.addGap(65)))
				.addComponent(checkBox))
			.addContainerGap())
		);
		gl_contentPane.setVerticalGroup(
			gl_contentPane.createParallelGroup(Alignment.LEADING)
				.addGroup(gl_contentPane.createSequentialGroup()
					.addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
						.addComponent(textField, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
						.addComponent(btnBrowse))
					.addPreferredGap(ComponentPlacement.RELATED)
					.addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
						.addComponent(textField_1, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
						.addComponent(btnAminofolder))
					.addPreferredGap(ComponentPlacement.RELATED)
					.addComponent(checkBox)
					.addPreferredGap(ComponentPlacement.RELATED, 6, Short.MAX_VALUE)
					.addComponent(scrollPane, GroupLayout.PREFERRED_SIZE, 324, GroupLayout.PREFERRED_SIZE)
					.addPreferredGap(ComponentPlacement.RELATED)
					.addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
						.addComponent(btnNucconv)
						.addComponent(btnProconv)
						.addComponent(btnReciprocal)
						.addComponent(btnNewButton)))
		);

		JTextArea textArea = new JTextArea();
		scrollPane.setViewportView(textArea);
		contentPane.setLayout(gl_contentPane);
	}
	private class SwingAction extends AbstractAction {
		public SwingAction() {
			putValue(NAME, "ヌクレオチド参照");
			putValue(SHORT_DESCRIPTION, "Some short description");
```

```
		}
		public void actionPerformed(ActionEvent e) {
			JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
			filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
			int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
			if (selected == JFileChooser.APPROVE_OPTION){
				textField.setText(filechooser.getSelectedFile().getAbsolutePath());
				/*
				File file = new File(textField_1.getText());
				String[] filelist = file.list();
				for (int i=0;i<filelist.length;i++){
					textArea.append(filelist[i]);
				}*/
			}
		}
}
private class SwingAction_1 extends AbstractAction {
		public SwingAction_1() {
			putValue(NAME, "アミノ酸参照");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
			filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
			int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
			if (selected == JFileChooser.APPROVE_OPTION){
				textField_1.setText(filechooser.getSelectedFile().getAbsolutePath());
				File file = new File(textField_1.getText());
				String[] filelist = file.list();

			}
		}
}
private class SwingAction_2 extends AbstractAction {
		public SwingAction_2() {
			putValue(NAME, "アミノ酸へ変換");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			try {
				ProteinCreator.nucToAmino(textField.getText(),checkBox.isSelected());
			} catch (FileNotFoundException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
		}
}
private class SwingAction_3 extends AbstractAction {
		public SwingAction_3() {
			putValue(NAME, "ヌクレオチドへ変換");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			try {
				NucConverter.aminoToNuc(textField_1.getText(),checkBox.isSelected());
			} catch (FileNotFoundException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
		}
}
private class SwingAction_4 extends AbstractAction {
		public SwingAction_4() {
			putValue(NAME, "双方向へ変換");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
		}
}
private class SwingAction_5 extends AbstractAction {
		public SwingAction_5() {
			putValue(NAME, "終始コドンを除く");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			try {
				ProteinCreator.removeStopcodon(textField.getText());
			} catch (FileNotFoundException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
```

```
                    }
                }
            }
```

Amino acid to nucleotide converter

```
package practice;
import javax.swing.*;
import javax.swing.text.DefaultEditorKit.*;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;
import org.biojava3.core.sequence.io.*;
import org.biojava3.core.sequence.*;

import java.awt.*;
import java.util.ArrayList;
import java.util.List;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class NucConverter extends JFrame implements ActionListener {
		private JTextArea textArea = new JTextArea();
		private JTextField species = new JTextField(10);

		//プログラム実行部分
		private JButton check = new JButton ("ヌクレオチド配列を保存");
		private JButton preOrtholog = new JButton("前回のプロファイルアラインメントフォルダ");
		private JFileChooser chooser = new JFileChooser("C:¥¥cDNAData");

		private ArrayList<FastaSequence> aafastaList;
		private ArrayList<FastaSequence> ntfastaList;
		private ArrayList<FastaSequence> ntGenomeFastaList;
		private ArrayList<String> doubledOrthologList = new ArrayList<String>();//重複した遺伝子名のリスト
		private String preOrthologAdress;
		private String genomeAdress = "C:¥¥cDNAData¥¥originalAlignments¥¥";
		private boolean preOrthologCheck = false;

		public NucConverter() {
			// TODO Auto-generated constructor stub
			super("ヌクレオチド配列データ作成");
			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			setSize(800,600);


			JPanel panel1 = new JPanel();
			JPanel panel2 = new JPanel();
			getContentPane().add(panel1,BorderLayout.NORTH);
			getContentPane().add(new JScrollPane(textArea),BorderLayout.CENTER);
			getContentPane().add(panel2,BorderLayout.SOUTH);

			panel1.add(preOrtholog);
			panel1.add(check);
			preOrtholog.addActionListener(this);
			check.addActionListener(this);

			chooser.setAcceptAllFileFilterUsed(false);
		}

		/**
		 * @param args
		 */
		public static void main(String[] args) {
			// TODO Auto-generated method stub
			new NucConverter().setVisible(true);

		}

		//[開く][保存]ボタンが押された時の処理
		public void actionPerformed(ActionEvent event){
			if (event.getSource().equals(check)){
				try {
					check();
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}else if (event.getSource().equals(preOrtholog)){
				try {
					openPreortholog();
				} catch (FileNotFoundException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
		}
```

```
}
//前回のプロファイルアラインメントと重なった遺伝子のデータの取得
private void openPreortholog() throws IOException {
                chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                int returnVal = chooser.showOpenDialog(this);
                if (returnVal == JFileChooser.APPROVE_OPTION){
                                File file = chooser.getSelectedFile();
                                preOrthologAdress =file.getAbsolutePath();
                                FileReader freader = new FileReader(preOrthologAdress + "¥¥protpreOrthologList.txt");
                                BufferedReader breader=new BufferedReader(freader);
                                String line;
                                while((line=breader.readLine())!=null){
                                   doubledOrthologList.add(line);
                                   System.out.println(line+":前回のプロファイルアラインメントと重複した遺伝子");
                                }
                                breader.close();
                                freader.close();
                                preOrthologCheck =true;
                }
}

public void check() throws FileNotFoundException,IOException {
                chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                int returnVal = chooser.showOpenDialog(this);
                if (returnVal == JFileChooser.APPROVE_OPTION){
                                File file = chooser.getSelectedFile();
                                setTitle(file.getAbsolutePath());
                                String fileAdress = file.getAbsolutePath();//変換したいアミノ酸を含むフォルダ
                                aminoToNuc(fileAdress,true);
                }
}
public static void aminoToNuc(String fileAdress,boolean profilealign) throws FileNotFoundException,IOException{
                File file = new File(fileAdress);
                String fileParentAdress = file.getParent();//アミノ酸の元のヌクレオチドを含むフォルダ
                File ntfile = new File(fileParentAdress);
                String aalist[] = file.list();
                String ntlist[] = ntfile.list();
    ArrayList<String> al = new ArrayList();
    ArrayList<String> nl = new ArrayList();
   String genomeAdress = "C:¥¥cDNAData¥¥originalAlignments¥¥";

    System.out.println("ファイルの読み取りまでは正常");

    for(int i=0;i<ntlist.length;i++){
                if (ntlist[i].indexOf(".fas")!=-1){
                                //ヌクレオチドリストの抽出
                                if (ntlist[i].indexOf("FBgn")!=-1){
                                                if (ntlist[i].indexOf("aligned_")==-1){
                                                                //textArea.append(ntlist[i]+"¥n");
                                                                nl.add(ntlist[i]);
                                                                /*
                                                                                if ((ntlist[i].indexOf(".fasta")!= -1)) {
                                                                                nl.add(ntlist[i]);
                                                                }
                                                                                else {
                                                                                nl.add(ntlist[i].replaceAll("fas","fasta"));
                                                                }*/
                                                }
                                }
                }
    }
    System.out.println("ヌクレオチドリストの作成成功");
    for(int j=0;j<aalist.length;j++){
                if (aalist[j].indexOf(".fas")!=-1){
                                //アライメント後のファイル名の抽出
                                if (aalist[j].indexOf("FBgn")!=-1){
                                                if ((aalist[j].indexOf("aligned_") != -1) || (aalist[j].indexOf("MultipleAligned_") != -1)){
                                                                al.add(aalist[j]);
                                                                /*
                                                                if ((aalist[j].indexOf(".fasta")!= -1)) {
                                                                                al.add(aalist[j]);
                                                                }
                                                                                else {
                                                                                al.add(aalist[j].replaceAll("fas","fasta"));
                                                                }*/
                                                }
                                }
                }
    }
    //アラインメント後のファイル読み込み
    if (al.size()==0){
                System.out.println("プロファイルアラインメント後のファイルがありません。");
    }

    //fasta 1 つずつについて処理
    int dup = 0;
    int dmod =0;
    for (int k=0;k<al.size();k++) {
                //アラインメント後のアミノ酸
          FileInputStream aainput = new FileInputStream(new File(fileAdress +"¥¥"+ al.get(k)));
```

```
List<FastaSequence> aafastaList = SequenceUtil.readFasta(aainput);
String regex = "FBgn.....";
Pattern pt = Pattern.compile(regex);
Matcher mt = pt.matcher(al.get(k));
String fastaName = null;
if(mt.find()){
	fastaName = mt.group();
}

//ヌクレオチドデータ（対象種+12+α）の作成
List<FastaSequence> ntfastaList = new ArrayList();
for (int l=0;l<nl.size();l++){
	if(nl.get(l).indexOf(fastaName)!=-1){
		FileInputStream ntinput = new FileInputStream(new File(fileParentAdress
+"¥¥"+ nl.get(l)));

		ntfastaList.addAll(SequenceUtil.readFasta(ntinput));

	}
}
File inputOrtholog = new File(genomeAdress + fastaName+".fasta");
if (inputOrtholog.exists() == false){
	inputOrtholog = new File(genomeAdress + fastaName+".fas");
}
FileInputStream nt12input=new FileInputStream(inputOrtholog);
List<FastaSequence> ntGenomeFastaList = SequenceUtil.readFasta(nt12input);

//以前にプロファイルアライメントした場合に適用
/*
if (preOrthologCheck == true) {
	for (int m=0;m<doubledOrthologList.size();m++) {
		if(al.get(k).equals(doubledOrthologList.get(m))){
			nt12input = new FileInputStream(new
File(preOrthologAdress +"¥¥"+ al.get(k)));
			ntGenomeFastaList =
(ArrayList<FastaSequence>)SequenceUtil.readFasta(nt12input);
		}
	}
}*/
if (profilealign){
	ntfastaList.addAll(ntGenomeFastaList);
}


//アミノ酸→ヌクレオチドのアルゴリズム
ArrayList<FastaSequence> ntTranslatedList = null;
//重複ヒットで使用
Boolean doublehit=false;
ArrayList<FastaSequence> newAAlist = null;
ArrayList<FastaSequence> newNTlist = null;

if (aafastaList.size() == ntfastaList.size()) {
	Collections.sort(aafastaList);
	Collections.sort(ntfastaList);
	ntTranslatedList = new ArrayList();
	for (int n=0;n<ntfastaList.size();n++){
		String aaseq = aafastaList.get(n).getSequence();
		String ntseq = ntfastaList.get(n).getSequence();
		StringBuilder sb = new StringBuilder(ntseq);
		for (int o=0;o<aaseq.length();o++){
			if (aaseq.charAt(o)=='-'){
				sb.insert(o*3, "---");
			}
		}
		FastaSequence ntFs = new
FastaSequence(ntfastaList.get(n).getId(),sb.toString());
		ntTranslatedList.add(ntFs);
	}
	//配列の長さをそろえる
	ntTranslatedList = (ArrayList<FastaSequence>) sortLength(ntTranslatedList);

	//複数ヒットコンティグの結合
	ArrayList<FastaSequence> aadouble = new ArrayList();
	ArrayList<FastaSequence> ntdouble = new ArrayList();
	String shareSp=null;
	for (int p=0;p<aafastaList.size();p++){
		shareSp = aafastaList.get(p).getId().substring(0,4);
		for (int q=p+1;q<aafastaList.size();q++){
			if (aafastaList.get(q).getId().indexOf(shareSp)!=-1){
				if (doublehit==false){

aadouble.add(aafastaList.get(p));

ntdouble.add(ntTranslatedList.get(p));
					doublehit=true;
				}
				aadouble.add(aafastaList.get(q));
				ntdouble.add(ntTranslatedList.get(q));
			}
		}
		if (doublehit){
```

```
                                                                    break;
                                                                }
                                                            }
                                                            if(doublehit){
                                                                //名前の結合
                                                                String contigId = "";
                                                                for (int i=0;i<aadouble.size();i++){
                                                                    contigId = contigId + " " +
FbgnFilter.contigfilter(aadouble.get(i).getId());
                                                                }

                                                                //アミノ酸配列の結合
                                                                dup++;
                                                                Boolean connectable =true;
                                                                String aaconnectSeq = aadouble.get(0).getSequence();
                                                                StringBuilder aacsb = new StringBuilder(aaconnectSeq);
                                                                for (int r=1;r<aadouble.size();r++){
                                                                    String aacompareSeq = aadouble.get(r).getSequence();
                                                                    for (int s=0;s<aaconnectSeq.length();s++){

            if(aaconnectSeq.charAt(s)!=aacompareSeq.charAt(s)){
                                                                        if (aaconnectSeq.charAt(s)=='-'){

            aacsb.setCharAt(s, aacompareSeq.charAt(s));
                                                                        }
                                                                        else
if(aaconnectSeq.charAt(s)=='X'){

            aacsb.setCharAt(s, aacompareSeq.charAt(s));
                                                                        }
                                                                        else{
                                                                            if
(aacompareSeq.charAt(s)=='-'){
                                                                            }
                                                                            else if
(aacompareSeq.charAt(s)=='X'){
                                                                            }
                                                                            else{

            connectable=false;

            break;
                                                                            }
                                                                        }
                                                                    }

                                                                }
                                                            }
                                                            //ヌクレオチド配列の結合
                                                            if(connectable){
                                                                String ntconnectSeq = ntdouble.get(0).getSequence();
                                                                StringBuilder ntcsb = new StringBuilder(ntconnectSeq);
                                                                for (int t=1;t<ntdouble.size();t++){
                                                                    String ntcompareSeq =
ntdouble.get(t).getSequence();

                                                                    for (int u=0;u<ntconnectSeq.length();u++){
                                                                        if
(ntconnectSeq.charAt(u)!=ntcompareSeq.charAt(u)){

            if(ntconnectSeq.charAt(u)=='-'){

            ntcsb.setCharAt(u, ntcompareSeq.charAt(u));
                                                                            }
                                                                            else
if(ntconnectSeq.charAt(u)=='N'){

            ntcsb.setCharAt(u, ntcompareSeq.charAt(u));
                                                                            }
                                                                            else if
(ntcompareSeq.charAt(u)=='-'){
                                                                            }
                                                                            else if
(ntcompareSeq.charAt(u)=='N'){
                                                                            }
                                                                            else {

            System.out.println(fastaName+":nt で結合できません");

            doublehit = false;

            break;
                                                                            }
                                                                        }

                                                                    }
                                                                }
                                                                //
                                                                newAAlist = new ArrayList();
                                                                newNTlist = new ArrayList();
```

```
                                                                        for (int v=0;v<aafastaList.size();v++){
                if(aafastaList.get(v).getId().indexOf(shareSp)!=-1){
                                                                                        }
                                                                                        else{
                newAAlist.add(aafastaList.get(v));
                newNTlist.add(ntTranslatedList.get(v));
                                                                                        }
                                                                        }
                                                                        shareSp = shareSp + "(" + contigId +")";
                                                                        newAAlist.add(new
FastaSequence(shareSp,aacsb.toString()));
                                                                        newNTlist.add(new FastaSequence(shareSp,ntcsb.toString()));
                                                                        Collections.sort(newNTlist);
                                                                }
                                                                else if(connectable == false) {
                                                                        doublehit = false;
                                                                        fastaName = "disconnected" + fastaName;
                                                                        System.out.println(fastaName+":"+shareSp+"で重なりが見つ
かりましたが、アミノ酸を結合できませんでした。");
                                                                        dmod++;
                                                                }

                                                        }

                                        }else{
                                                System.out.println(fastaName+"ヌクレオチドとアミノ酸の格納数が一致しません。アミノ
酸:"+aafastaList.size()+"ヌクレオチド:"+ntfastaList.size());
                                        }

                                        String outfilename = fileParentAdress +"¥¥aligned_"+ fastaName + ".fasta";
                                        OutputStream os = new FileOutputStream(outfilename);
                                        if (doublehit){
                                                org.biojava3.data.sequence.SequenceUtil.writeFasta(os, newNTlist);
                                        }
                                        else {
                                                org.biojava3.data.sequence.SequenceUtil.writeFasta(os, ntTranslatedList);
                                        }
                                        os.close();

                }
                System.out.println("同一遺伝子内での結合試行回数:"+dup);
                System.out.println("同一遺伝子内での結合失敗回数:"+dmod);
        }
        private static List<FastaSequence> sortLength(List<FastaSequence> ntTranslatedList) {
                        Collections.sort(ntTranslatedList,new FastaComparator());
                        int length0 = ntTranslatedList.get(0).getLength();
                        ArrayList<FastaSequence> lengthmod = new ArrayList();
                        lengthmod.add(new FastaSequence(ntTranslatedList.get(0).getId(),ntTranslatedList.get(0).getSequence()));
                        for (int x=1;x<ntTranslatedList.size();x++){
                                int length1 = ntTranslatedList.get(x).getLength();
                                String modseq = null;
                                if(length0<length1){
                                        modseq = ntTranslatedList.get(x).getSequence().substring(0,length0);
                                }
                                else {
                                        modseq = ntTranslatedList.get(x).getSequence();
                                }
                                FastaSequence mfs = new FastaSequence(ntTranslatedList.get(x).getId(),modseq);
                                lengthmod.add(mfs);
                        }
                        ntTranslatedList = lengthmod;
                        Collections.sort(ntTranslatedList);
                        return ntTranslatedList;
        }
}
```

Nucleotide to amino acid converter

```
package practice;
import javax.swing.*;
import javax.swing.text.DefaultEditorKit.*;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;
import org.biojava3.core.sequence.io.*;
import org.biojava3.core.sequence.*;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


@SuppressWarnings("serial")
public class ProteinCreator extends JFrame implements ActionListener{

		private JTextArea textArea = new JTextArea();
		private JTextField species = new JTextField(10);

		//プログラム実行部分
		private JButton check = new JButton ("アミノ酸配列を保存");
		private JFileChooser chooser = new JFileChooser("C:¥¥cDNAData");

		private ArrayList<FastaSequence> fastaList = new ArrayList<FastaSequence>();
		private ArrayList<FastaSequence> fastaList3 = new ArrayList<FastaSequence>();//最後に、NGS データのみをまとめたリスト

		//コンストラクタ
		public ProteinCreator(){
			super("アミノ酸配列データ作成");
			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			setSize(800,600);


			JPanel panel1 = new JPanel();
			JPanel panel2 = new JPanel();
			getContentPane().add(panel1,BorderLayout.NORTH);
			getContentPane().add(new JScrollPane(textArea),BorderLayout.CENTER);
			getContentPane().add(panel2,BorderLayout.SOUTH);

			panel1.add(check);
			check.addActionListener(this);

			chooser.setAcceptAllFileFilterUsed(false);

		}

		public static void main(String[] args) {
			new ProteinCreator().setVisible(true);

		}

		//[開く][保存]ボタンが押された時の処理
		public void actionPerformed(ActionEvent event){
			if (event.getSource().equals(check)){
				try {
					check();
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}

		}

		//ファイルを保存するメソッド
		private void save() {
			int returnVal = chooser.showSaveDialog(this);
			try {
				if (returnVal == JFileChooser.APPROVE_OPTION) {
					FileWriter fWriter = new FileWriter(chooser.getSelectedFile());
					BufferedWriter bWriter = new BufferedWriter(fWriter);
					bWriter.write(textArea.getText());
					bWriter.flush();
					fWriter.close();
					bWriter.close();

				}
			} catch (FileNotFoundException event) {
				event.printStackTrace();
			} catch (IOException event) {
				event.printStackTrace();
```

```
		}
}

//check メソッド
private void check() throws FileNotFoundException,IOException {
			chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
				int returnVal = chooser.showOpenDialog(this);
				boolean profilealign = true;
				if (returnVal == JFileChooser.APPROVE_OPTION){
					File file = chooser.getSelectedFile();
					setTitle(file.getAbsolutePath());
					String fileAdress = file.getAbsolutePath();
					nucToAmino(fileAdress, profilealign);
				}
}
public static void nucToAmino(String fileAdress,boolean profilealign) throws FileNotFoundException,IOException{
		File file = new File(fileAdress);
		String fileParentAdress = file.getParent();
		String ntlist[] = file.list();
	ArrayList al = new ArrayList();
	for(int j=0;j<ntlist.length;j++){
		if (ntlist[j].indexOf(".fas")!= -1){
			//textArea.append(ntlist[j]+"¥n");
			al.add(ntlist[j]);
			/*
				if ((ntlist[j].indexOf(".fasta")!= -1)) {
				al.add(ntlist[j]);
			}
				else {
				al.add(ntlist[j].replaceAll("fas","fasta"));
			}*/
		}
	}
	int count = 0;
	for (int i=0; i<al.size(); i++) {
			String infilename = fileAdress +"¥¥"+ al.get(i);
		FileInputStream input = new FileInputStream(infilename);
			List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
			input.close();
			ArrayList<FastaSequence> proList = new ArrayList<FastaSequence>();

			for (int j=0;j<fastaList.size();j++){
				String seq = fastaList.get(j).getSequence().toString();
				String name = fastaList.get(j).getId();
				//アミノ酸シーケンスの作成
				DNASequence ds = new DNASequence(seq);
				RNASequence rs = ds.getRNASequence();
				ProteinSequence ps = rs.getProteinSequence();
				String amino = ps.getSequenceAsString();
				if (profilealign){
					amino = amino.replaceAll("X","");
				}
				FastaSequence proFs = new FastaSequence(name,amino);
				proList.add(proFs);
			}
			String outfilename = fileAdress + "¥¥prot¥¥" + al.get(i);
			OutputStream os = new FileOutputStream(outfilename);
		SequenceUtil.writeFasta(os, proList);
			os.close();
			count++;
	}
}
public static void removeStopcodon(String fileAdress) throws FileNotFoundException,IOException{
		File file = new File(fileAdress);
		String fileParentAdress = file.getParent();
		String ntlist[] = file.list();
	ArrayList al = new ArrayList();
	for(int j=0;j<ntlist.length;j++){
		if (ntlist[j].indexOf(".fas")!= -1){
			al.add(ntlist[j]);
		}
	}
	int count = 0;
	for (int i=0; i<al.size(); i++) {
			String infilename = fileAdress +"¥¥"+ al.get(i);
		FileInputStream input = new FileInputStream(infilename);
			List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
			input.close();
			ArrayList<FastaSequence> modList = new ArrayList<FastaSequence>();

			for (int j=0;j<fastaList.size();j++){
				String seq = fastaList.get(j).getSequence().toString();
				String name = fastaList.get(j).getId();
				//アミノ酸シーケンスの作成
				DNASequence ds = new DNASequence(seq);
				RNASequence rs = ds.getRNASequence();
				ProteinSequence ps = rs.getProteinSequence();
				String amino = ps.getSequenceAsString();
				StringBuilder sb = new StringBuilder(seq);
				int nstop = 0;
```

```
                                        for (int k=0;k<amino.length();k++){
                                                if ((amino.charAt(k)=='*')){
                                                        int s = (k-nstop)*3-1;
                                                        int e = (k-nstop)*3+2;
                                                        if (s<0){
                                                                s = 0;
                                                                e = 3;
                                                        }
                                                        sb.delete(s, e);
                                                        sb.append("---");
                                                        nstop++;
                                                }
                                        }
                                        FastaSequence modFs = new FastaSequence(name,sb.toString());
                                        modList.add(modFs);
                                }
                                String outfilename = fileAdress + "¥¥stopRemoved_" + al.get(i);
                                OutputStream os = new FileOutputStream(outfilename);
                        SequenceUtil.writeFasta(os, modList);
                                os.close();
                                count++;
                        }
                }
        }
```

Fasta comparator

```
package practice;

import java.util.Comparator;
import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;
import org.biojava3.core.sequence.io.*;
import org.biojava3.core.sequence.*;


//シーケンスの長さでソート
public class FastaComparator implements Comparator<FastaSequence>{

		public FastaComparator() {
			// TODO Auto-generated constructor stub
		}
		public int compare(FastaSequence f0, FastaSequence f1) {
			// TODO Auto-generated method stub
			int l1=f0.getLength();
			int l2=f1.getLength();

			if(l1>l2){
				return 1;
			}else if(l1<l2){
				return -1;
			}else {
				return 0;
			}
		}

}
```

To merge all fasta files to one fasta file in same folder for reciprocal blast

```
package practice;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JFileChooser;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

import javax.swing.AbstractAction;
import javax.swing.Action;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;
import javax.swing.JLabel;

public class MakeFastaDB extends JFrame {

	private JPanel contentPane;
	private JTextField txtSetfolder;
	private final Action action = new SwingAction();
	private final Action action_1 = new SwingAction_1();
	private JTextField textField;
	private final Action action_2 = new SwingAction_2();
	private JTextField textField_1;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					MakeFastaDB frame = new MakeFastaDB();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public MakeFastaDB() {
		setTitle("Fasta¥u30D5¥u30A1¥u30A4¥u30EB¥u306E¥u4E00¥u62EC¥u5316");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 501, 425);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		JScrollPane scrollPane = new JScrollPane();

		txtSetfolder = new JTextField();
		txtSetfolder.setText("setFolder");
		txtSetfolder.setColumns(10);

		JButton btnSetfolder = new JButton("参照");
		btnSetfolder.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
			}
		});
		btnSetfolder.setAction(action);

		JButton btnNewButton = new JButton("New button");
```

```
                btnNewButton.setAction(action_1);
                btnNewButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                        }
                });

                textField = new JTextField();
                textField.setColumns(10);

                JButton btnNotortho = new JButton("notOrtho");
                btnNotortho.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                        }
                });
                btnNotortho.setAction(action_2);

                JLabel label = new JLabel("¥u7A2E¥u540D");

                textField_1 = new JTextField();
                textField_1.setColumns(10);
                GroupLayout gl_contentPane = new GroupLayout(contentPane);
                gl_contentPane.setHorizontalGroup(
                        gl_contentPane.createParallelGroup(Alignment.TRAILING)
                                .addGroup(gl_contentPane.createSequentialGroup()
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                .addGroup(Alignment.TRAILING,
gl_contentPane.createSequentialGroup()
                                                        .addGap(164)
                                                        .addComponent(btnNewButton,
GroupLayout.DEFAULT_SIZE, 142, Short.MAX_VALUE)
                                                        .addGap(157))
                                                .addGroup(Alignment.TRAILING,
gl_contentPane.createSequentialGroup()
                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                        .addComponent(textField,
GroupLayout.DEFAULT_SIZE, 341, Short.MAX_VALUE)
                                                        .addComponent(txtSetfolder,
Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, 341, Short.MAX_VALUE))
                .addPreferredGap(ComponentPlacement.RELATED)
                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                        .addComponent(btnSetfolder,
GroupLayout.DEFAULT_SIZE, 115, Short.MAX_VALUE)
                                                        .addComponent(btnNotortho,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                                                .addComponent(scrollPane, Alignment.TRAILING,
GroupLayout.DEFAULT_SIZE, 463, Short.MAX_VALUE)
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addComponent(label,
GroupLayout.PREFERRED_SIZE, 36, GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(ComponentPlacement.RELATED)
                                                        .addComponent(textField_1,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)))
                                                .addContainerGap())
                        );
                        gl_contentPane.setVerticalGroup(
                                gl_contentPane.createParallelGroup(Alignment.LEADING)
                                        .addGroup(gl_contentPane.createSequentialGroup()
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(txtSetfolder,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(btnSetfolder))
                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                        .addGroup(gl_contentPane.createSequentialGroup()
                                                                .addGap(2)
                                                                .addComponent(textField,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                                        .addComponent(btnNotortho))
                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(label, GroupLayout.PREFERRED_SIZE, 16,
GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(textField_1,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                                .addGap(18)
                                                .addComponent(scrollPane, GroupLayout.DEFAULT_SIZE, 262,
Short.MAX_VALUE)
                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                .addComponent(btnNewButton))
                        );

                        JTextArea textArea = new JTextArea();
                        scrollPane.setViewportView(textArea);
                        contentPane.setLayout(gl_contentPane);
                }
                private class SwingAction extends AbstractAction {
                        public SwingAction() {
```

```
                putValue(NAME, "参照");
                putValue(SHORT_DESCRIPTION, "Some short description");
            }
            public void actionPerformed(ActionEvent e) {
                JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
                filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                if (selected == JFileChooser.APPROVE_OPTION){
                    txtSetfolder.setText(filechooser.getSelectedFile().getAbsolutePath());
                }
            }
        }
        private class SwingAction_1 extends AbstractAction {
            public SwingAction_1() {
                putValue(NAME, "DB 用 fasta の生成");
                putValue(SHORT_DESCRIPTION, "Some short description");
            }
            public void actionPerformed(ActionEvent e) {
                try{
                    String fileAd = txtSetfolder.getText();
                    File file = new File(fileAd);
                    File[] fastalist = file.listFiles(new FastaNameFilter());
                    String[] namelist = file.list(new FastaNameFilter());
                List<FastaSequence> dbfastalist = new ArrayList();

                for (int i=0;i<fastalist.length;i++){
                    FileInputStream input = new FileInputStream(fastalist[i]);
                    List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
                    input.close();
                    //種名にあったシーケンスリストの作成
                    String spname = textField_1.getText();
                    for (int j=0;j<fastaList.size();j++){
                        if (fastaList.get(j).getId().indexOf(spname) != -1){
                            String id = FbgnFilter.fbgnfilter(namelist[i]) + " " +
FbgnFilter.parenthesisfilter(fastaList.get(j).getId());
                            dbfastalist.add(new
FastaSequence(id,fastaList.get(j).getSequence()));
                        }
                    }
                }
                String outfilename = fileAd + "¥¥allOrtholog.fasta";
                OutputStream os = new FileOutputStream(outfilename);
                SequenceUtil.writeFasta(os, dbfastalist);
                    os.close();

                    String notOrthoAd = textField.getText();
                    FileInputStream input = new FileInputStream(notOrthoAd);
                List<FastaSequence> notOrthoList = SequenceUtil.readFasta(input);
                input.close();
                notOrthoList.addAll(dbfastalist);
                List<FastaSequence> allcontigList = new ArrayList();
                for (int i=0;i<notOrthoList.size();i++){
                    String seq = notOrthoList.get(i).getSequence();
                    seq = seq.replaceAll("¥¥*","");
                    seq = seq.replaceAll("-","");
                    seq = seq.replaceAll("X","");
                    seq = seq.replaceAll("¥¥?","");
                    allcontigList.add(new FastaSequence(notOrthoList.get(i).getId(),seq));
                }

                String allcontigAd = fileAd + "¥¥allcontigsMod.fasta";
                os = new FileOutputStream(allcontigAd);
                SequenceUtil.writeFasta(os, allcontigList);
                    os.close();
                } catch (IOException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }

            }
        }
        private class SwingAction_2 extends AbstractAction {
            public SwingAction_2() {
                putValue(NAME, "非オーソログ fasta");
                putValue(SHORT_DESCRIPTION, "Some short description");
            }
            public void actionPerformed(ActionEvent e) {
                JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
                filechooser.setFileFilter(new FastaFilter());
                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                if (selected == JFileChooser.APPROVE_OPTION){
                    textField.setText(filechooser.getSelectedFile().getAbsolutePath());
                }
            }
        }
}
```

To move orthologous gene fasta files passed reciprocal blast

```
package practice;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JFileChooser;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.swing.Action;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;

import javax.swing.JLabel;

public class ReciprocalBlast extends JFrame {

	private JPanel contentPane;
	private JTextField txtSetfolder;
	private JTextArea textArea;
	private ArrayList<String> reciprocal;
	private final Action action = new SwingAction();
	private final Action action_1 = new SwingAction_1();
	private JTextField txtSetsaveorthologfolder;
	private JButton btnNewButton_1;
	private final Action action_2 = new SwingAction_2();
	private JTextField txtDste;
	private JButton btnSave;
	private final Action action_3 = new SwingAction_3();
	private JScrollPane scrollPane_1;
	private JTextArea textArea_1;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					ReciprocalBlast frame = new ReciprocalBlast();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public ReciprocalBlast() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 566, 480);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		txtSetfolder = new JTextField();
		txtSetfolder.setText("setOutFile");
		txtSetfolder.setColumns(10);

		JButton btnNewButton = new JButton("New button");
```

```
                btnNewButton.setAction(action);

                JScrollPane scrollPane = new JScrollPane();

                JButton btnCheck = new JButton("check");
                btnCheck.setAction(action_1);

                txtSetsaveorthologfolder = new JTextField();
                txtSetsaveorthologfolder.setText("setSaveOrthologFolder");
                txtSetsaveorthologfolder.setColumns(10);

                btnNewButton_1 = new JButton("New button");
                btnNewButton_1.setAction(action_2);

                JLabel label = new JLabel("¥u7A2E¥u540D");

                txtDste = new JTextField();
                txtDste.setText("dste");
                txtDste.setColumns(10);

                btnSave = new JButton("save");
                btnSave.setAction(action_3);

                scrollPane_1 = new JScrollPane();
                GroupLayout gl_contentPane = new GroupLayout(contentPane);
                gl_contentPane.setHorizontalGroup(
                        gl_contentPane.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_contentPane.createSequentialGroup()
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
                                                .addComponent(txtSetsaveorthologfolder,
Alignment.LEADING, GroupLayout.DEFAULT_SIZE, 408, Short.MAX_VALUE)
                                                .addComponent(txtSetfolder,    GroupLayout.DEFAULT_SIZE,
408, Short.MAX_VALUE))
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING, false)
                                                .addComponent(btnNewButton_1,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                .addComponent(btnNewButton,
GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                                        .addContainerGap())
                                .addGroup(gl_contentPane.createSequentialGroup()
                                        .addComponent(label)
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addComponent(txtDste,                GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                        .addGap(299))
                                .addGroup(Alignment.TRAILING, gl_contentPane.createSequentialGroup()
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
                                                .addComponent(scrollPane,                Alignment.LEADING,
GroupLayout.DEFAULT_SIZE, 262, Short.MAX_VALUE)
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addGap(41)
                                                        .addComponent(btnCheck,
GroupLayout.DEFAULT_SIZE, 221, Short.MAX_VALUE)))
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addComponent(btnSave)
                                                        .addGap(46))
                                                .addGroup(gl_contentPane.createSequentialGroup()
                                                        .addComponent(scrollPane_1,
GroupLayout.PREFERRED_SIZE, 259, GroupLayout.PREFERRED_SIZE)
                                                        .addContainerGap()))));
                gl_contentPane.setVerticalGroup(
                        gl_contentPane.createParallelGroup(Alignment.LEADING)
                                .addGroup(gl_contentPane.createSequentialGroup()
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(txtSetfolder,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addComponent(btnNewButton))
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(txtSetsaveorthologfolder,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                .addComponent(btnNewButton_1))
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(label)
                                                .addComponent(txtDste,    GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
                                        .addPreferredGap(ComponentPlacement.RELATED, 43, Short.MAX_VALUE)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                                                .addComponent(scrollPane_1)
                                                .addComponent(scrollPane,                Alignment.TRAILING,
GroupLayout.PREFERRED_SIZE, 289, GroupLayout.PREFERRED_SIZE))
                                        .addPreferredGap(ComponentPlacement.RELATED)
                                        .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                .addComponent(btnCheck)
                                                .addComponent(btnSave)))
                );
```

```
		textArea_1 = new JTextArea();
		scrollPane_1.setViewportView(textArea_1);

		textArea = new JTextArea();
		scrollPane.setViewportView(textArea);
		contentPane.setLayout(gl_contentPane);
	}
	private class SwingAction extends AbstractAction {
		public SwingAction() {
			putValue(NAME, "out ファイル参照");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
			filechooser.setFileFilter(new OutFilter());
			int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
			if (selected == JFileChooser.APPROVE_OPTION){
				txtSetfolder.setText(filechooser.getSelectedFile().getAbsolutePath());
			}
		}
	}
	private class SwingAction_1 extends AbstractAction {
		public SwingAction_1() {
			putValue(NAME, "双方向ヒットの配列リストを探す");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			try {
				reciprocal = checkReciprocal(txtSetfolder.getText());
				for (int i=0;i<reciprocal.size();i++){
					textArea.append(reciprocal.get(i) + "¥n");
				}
				int count = reciprocal.size();
				textArea.append("遺伝子数："+ count);

			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
		}
	}
	private ArrayList<String> checkReciprocal(String fileAd) throws IOException {
		FileReader fReader;
		fReader = new FileReader(fileAd);
		BufferedReader bReader = new BufferedReader(fReader);

		ArrayList<String> recip = new ArrayList();
		String line = "start";
		boolean indelcheck = false;
		int hitCount = 0;
		while(line.indexOf("Number of sequences in database:") == -1) {
			line = bReader.readLine();
			String query = "";
			// 1 つのクエリーについて読み始める
			if(line.indexOf("parent=")!=-1){
				query = FbgnFilter.fbgnfilter(line);
		// 1 つの遺伝子内について、情報を集める
				String geneId = null;

			while ( line.indexOf("Effective search space used:") == -1 ){
				line = bReader.readLine();
					//ブラストヒットした場合の処理
					if(line.indexOf("Sequences producing significant alignments") != -1){
						while ( line.indexOf("Effective search space used:") == -1 ){
							line = bReader.readLine();
							//ブラストヒットの一番最初の遺伝子についての
読み取り
							if(line.indexOf("FBgn") != -1){
								geneId =
FbgnFilter.fbgnfilter(line);
								break;
							}
						}
					}
					//ブラストヒットしなかった場合
					else if(line.indexOf("No hits found")!=-1){

					}

			}
			//遺伝子内情報を集めた後、seqData の生成
			if(line.indexOf("Effective search space used:") != -1){
				if (query.equals(geneId)){
					recip.add(geneId);
					geneId = null;
				}

			}
		}
```

```
                }
                bReader.close();
                fReader.close();
                Collections.sort(recip);
                ArrayList<String> modlist = new ArrayList();
                for (int i=0;i<recip.size()-1;i++){
                        if (recip.get(i).equals(recip.get(i+1)) == false){
                                modlist.add(recip.get(i));
                                if(i == recip.size()-1){
                                        modlist.add(recip.get(i+1));
                                }
                        }

                }
                return modlist;
        }
        private class SwingAction_2 extends AbstractAction {
                public SwingAction_2() {
                        putValue(NAME, "オーソログフォルダ");
                        putValue(SHORT_DESCRIPTION, "Some short description");
                }
                public void actionPerformed(ActionEvent e) {
                        JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
                        filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                        int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                        if (selected == JFileChooser.APPROVE_OPTION){
                                txtSetsaveorthologfolder.setText(filechooser.getSelectedFile().getAbsolutePath());
                        }
                }
        }
        /*
        private void SaveReciprocal(String savead,ArrayList<String> reciprocal,String spname) throws IOException{
                File file = new File(savead);
                String[] orthologlist = file.list(new FastaNameFilter());

            for (int i=0;i<orthologlist.length;i++){
                for (int j=0;j<reciprocal.size();j++){
                        if (orthologlist[i].indexOf(reciprocal.get(j))!= -1){
                                System.out.println(orthologlist[i] + "一致しました。");
                                File copyfasta = new File(savead + "¥¥" + orthologlist[i]);
                        FileInputStream input = new FileInputStream(copyfasta);
                                List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
                                input.close();

                                List<FastaSequence> reciprocallist = new ArrayList();
                                //種名にあったシーケンスリストの作成
                        for (int k=0;k<fastaList.size();j++){
                                if (fastaList.get(k).getId().indexOf(spname) != -1){
                                        reciprocallist.add(fastaList.get(k));
                                }
                        }

                        File target = new File(savead + "¥¥reciprocal¥¥" + orthologlist[i]);
                        FileOutputStream output = new FileOutputStream(target);
                        SequenceUtil.writeFasta(output, reciprocallist);
                        output.close();
                        }
                }
            }
        }*/
        private class SwingAction_3 extends AbstractAction {
                public SwingAction_3() {
                        putValue(NAME, "レシプロカルの移動スクリプト");
                        putValue(SHORT_DESCRIPTION, "Some short description");
                }
                public void actionPerformed(ActionEvent e) {
                        moveScript(txtSetsaveorthologfolder.getText(),reciprocal);
                }
        }
        private void moveScript (String savead,ArrayList<String> reciprocal){
                File file = new File(savead);
                String[] orthologlist = file.list(new FastaNameFilter());

            for (int i=0;i<orthologlist.length;i++){
                for (int j=0;j<reciprocal.size();j++){
                        if (orthologlist[i].indexOf(reciprocal.get(j))!= -1){
                                System.out.println(orthologlist[i] + "一致しました。");
                                textArea_1.append("move " + orthologlist[i] + " "+ savead + "¥¥reciprocal" + "¥n");
                        }
                }
            }
        }
}
```

Distance matrix maker for check alignment

```
package practice;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Map;

import javax.swing.Action;

import org.biojava3.core.sequence.*;
import org.biojava3.core.sequence.storage.BitSequenceReader;
import org.biojava3.core.sequence.io.FastaReaderHelper;
import org.biojava3.data.sequence.*;
import org.biojava3.alignment.*;
import org.biojava3.phylo.*;

import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import java.awt.event.ActionListener;

public class DistanceMatrixMaker extends JFrame {

	private JPanel contentPane;
	private JTextField textField;
	private final Action action = new SwingAction();
	private JTextField textField_1;
	private final Action action_1 = new SwingAction_1();
	JTextArea txtrAlignscore;
	JTextArea textArea;
	JTextArea textArea_1;

	/**
	 * Launch the application.
	 */
	private ArrayList<FastaSequence> fastaList;
	private final Action action_2 = new SwingAction_2();

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					DistanceMatrixMaker frame = new DistanceMatrixMaker();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public DistanceMatrixMaker() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 535, 772);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		JButton btnSetfolder = new JButton("setFolder");
		btnSetfolder.setAction(action);

		textField = new JTextField();
		textField.setColumns(10);
```

```
		textField_1 = new JTextField();
		textField_1.setColumns(10);

		JButton btnAminodistance = new JButton("aminoDistance");
		btnAminodistance.setAction(action_1);

		JScrollPane scrollPane = new JScrollPane();

		JScrollPane scrollPane_1 = new JScrollPane();

		JScrollPane scrollPane_2 = new JScrollPane();
		GroupLayout gl_contentPane = new GroupLayout(contentPane);
		gl_contentPane.setHorizontalGroup(
			gl_contentPane.createParallelGroup(Alignment.LEADING)
				.addGroup(gl_contentPane.createSequentialGroup()
					.addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
						.addComponent(textField_1, Alignment.LEADING, GroupLayout.DEFAULT_SIZE, 329, Short.MAX_VALUE)
						.addComponent(textField, GroupLayout.DEFAULT_SIZE, 329, Short.MAX_VALUE))
					.addPreferredGap(ComponentPlacement.RELATED)
					.addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
						.addComponent(btnSetfolder, GroupLayout.DEFAULT_SIZE, 173, Short.MAX_VALUE)
						.addComponent(btnAminodistance, GroupLayout.DEFAULT_SIZE, 173, Short.MAX_VALUE)))
				.addComponent(scrollPane, GroupLayout.DEFAULT_SIZE, 509, Short.MAX_VALUE)
				.addComponent(scrollPane_1, GroupLayout.DEFAULT_SIZE, 509, Short.MAX_VALUE)
				.addComponent(scrollPane_2, GroupLayout.DEFAULT_SIZE, 509, Short.MAX_VALUE)
		);
		gl_contentPane.setVerticalGroup(
			gl_contentPane.createParallelGroup(Alignment.LEADING)
				.addGroup(gl_contentPane.createSequentialGroup()
					.addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
						.addComponent(btnSetfolder)
						.addComponent(textField, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
					.addPreferredGap(ComponentPlacement.RELATED)
					.addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
						.addComponent(textField_1, GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
						.addComponent(btnAminodistance))
					.addPreferredGap(ComponentPlacement.RELATED)
					.addComponent(scrollPane, GroupLayout.PREFERRED_SIZE, 217, GroupLayout.PREFERRED_SIZE)
					.addPreferredGap(ComponentPlacement.RELATED)
					.addComponent(scrollPane_1, GroupLayout.PREFERRED_SIZE, 217, GroupLayout.PREFERRED_SIZE)
					.addPreferredGap(ComponentPlacement.RELATED)
					.addComponent(scrollPane_2, GroupLayout.DEFAULT_SIZE, 224, Short.MAX_VALUE))
		);

		textArea_1 = new JTextArea();
		scrollPane_2.setViewportView(textArea_1);

		textArea = new JTextArea();
		scrollPane_1.setViewportView(textArea);

		txtrAlignscore = new JTextArea();
		scrollPane.setViewportView(txtrAlignscore);
		contentPane.setLayout(gl_contentPane);
	}
	private class SwingAction extends AbstractAction {
		public SwingAction() {
			putValue(NAME, "ヌクレオチド距離");
			putValue(SHORT_DESCRIPTION, "Some short description");
		}
		public void actionPerformed(ActionEvent e) {
			JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
			filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
			int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
			try {
				if (selected == JFileChooser.APPROVE_OPTION){
					File file = filechooser.getSelectedFile();
					String fileAdress = file.getAbsolutePath();
					textField.setText(fileAdress);
					String ntlist[] = file.list();
				ArrayList nl = new ArrayList();
				for(int i=0;i<ntlist.length;i++){
					if (ntlist[i].indexOf(".fas")!= -1){
						nl.add(ntlist[i]);
					}
				}
				ArrayList scoreList = new ArrayList();
				for (int n=0;n<nl.size();n++){
					String infilename = fileAdress +"¥¥"+ nl.get(n);
				FileInputStream input = new FileInputStream(infilename);
					List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
					input.close();
```

```
                                                List<FastaSequence> genomeList = takeGenomes(fastaList);
                                                int seqlist[][] = createNTSeqMat(fastaList);
                                                        double pdistanceMatrix[][] = computePDistance(seqlist);
                                                        int genomeSeqlist[][] = createNTSeqMat(genomeList);
                                                    double genomeDistance[][] = computePDistance(genomeSeqlist);
                                                        double                                  score                                   =
testAlignment(pdistanceMatrix,genomeDistance);
                                                        scoreList.add(new ScoreData((String)nl.get(n),score));
                                                }
                                                //Collections.sort(scoreList,new ScoreComparator());
                                                Collections.sort(scoreList);
                                                for (int i=0;i<scoreList.size();i++){
                                                        txtrAlignscore.append(((ScoreData)
scoreList.get(i)).getNameScore()+"¥n");
                                                }
                                        }
                                } catch (FileNotFoundException event) {
                                        event.printStackTrace();
                                } catch (IOException event) {
                                        event.printStackTrace();
                                } catch (Exception e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                }
                        }

        }
        private class SwingAction_1 extends AbstractAction {
                public SwingAction_1() {
                        putValue(NAME, "アミノ酸距離");
                        putValue(SHORT_DESCRIPTION, "Some short description");
                }
                public void actionPerformed(ActionEvent e) {
                        JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
                        filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                        int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                        try {
                                if (selected == JFileChooser.APPROVE_OPTION){
                                        File file = filechooser.getSelectedFile();
                                        String fileAdress = file.getAbsolutePath();
                                        textField_1.setText(fileAdress);
                                        String aalist[] = file.list();
                                ArrayList al = new ArrayList();
                                for(int i=0;i<aalist.length;i++){
                                        if (aalist[i].indexOf(".fas")!= -1){
                                                al.add(aalist[i]);
                                        }
                                }

                                ArrayList scoreList = new ArrayList();
                                ArrayList paralogScoreList = new ArrayList();
                                ArrayList stopCodonList = new ArrayList();
                                for (int n=0;n<al.size();n++){
                                        String infilename = fileAdress +"¥¥"+ al.get(n);
                                    FileInputStream input = new FileInputStream(infilename);
                                        List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
                                        input.close();
                                        List<FastaSequence> genomeList = takeGenomes(fastaList);
                                            //全てのディスタンスを計算
                                            int seqlist[][] = createAASeqMat(fastaList);
                                            double pdistanceMatrix[][] = computePDistance(seqlist);
                                            //12 ゲノムのみのディスタンスを計算
                                            int genomeSeqlist[][] = createAASeqMat(genomeList);
                                            double genomeDistance[][] = computePDistance(genomeSeqlist);
                                            double score = testAlignment(pdistanceMatrix,genomeDistance);
                                            scoreList.add(new ScoreData((String)al.get(n),score));
                                            //パラログチェック
                                            double paraScore = getParalogScore(seqlist,genomeSeqlist);
                                            paralogScoreList.add(new ScoreData((String)al.get(n),paraScore));
                                            //stop codon カウント
                                            int stopCodon = getStopCodon(seqlist);
                                            stopCodonList.add(new ScoreData((String)al.get(n),stopCodon));
                                }
                        Collections.sort(scoreList,new ScoreComparator());
                        Collections.sort(paralogScoreList,new ScoreComparator());
                        Collections.sort(stopCodonList,new ScoreComparator());
                        for (int i=0;i<scoreList.size();i++){
                                        txtrAlignscore.append(((ScoreData) scoreList.get(i)).getNameScore()+"¥n");
                                        textArea.append(((ScoreData)
paralogScoreList.get(i)).getNameScore()+"¥n");
                                        textArea_1.append(((ScoreData) stopCodonList.get(i)).getNameScore()+"¥n");
                        }
                                }
                                } catch (FileNotFoundException event) {
                                        event.printStackTrace();
                                } catch (IOException event) {
                                        event.printStackTrace();
                                } catch (Exception e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
```

```
                        }
                }
}
public int[][] createNTSeqMat(List<FastaSequence> fastalist){
                Collections.sort(fastalist,new FastaComparator());
                Collections.reverse(fastalist);
                int seqlist[][] = new int[fastalist.size()][fastalist.get(0).getLength()];
                for (int i=0;i<seqlist.length;i++){
                        String seq = fastalist.get(i).getSequence();
                        for (int j=0;j<seq.length();j++){
                                switch(seq.charAt(j)){
                                case 'A':seqlist[i][j] = 1;break;
                                case 'G':seqlist[i][j] = 2;break;
                                case 'C':seqlist[i][j] = 3;break;
                                case 'T':seqlist[i][j] = 4;break;
                                case '-':seqlist[i][j] = 0;break;
                                case 'N':seqlist[i][j] = 0;break;
                                default: seqlist[i][j] = 0;break;
                                }
                        }
                }
                return seqlist;
}
public int[][] createAASeqMat(List<FastaSequence> fastalist){
                Collections.sort(fastalist,new FastaComparator());
                Collections.reverse(fastalist);
                int seqlist[][] = new int[fastalist.size()][fastalist.get(0).getLength()];
                for (int i=0;i<seqlist.length;i++){
                        String seq = fastalist.get(i).getSequence();
                        for (int j=0;j<seq.length();j++){
                                switch(seq.charAt(j)){
                                case 'A':seqlist[i][j] = 1;break;
                                case 'C':seqlist[i][j] = 2;break;
                                case 'D':seqlist[i][j] = 3;break;
                                case 'E':seqlist[i][j] = 4;break;
                                case 'F':seqlist[i][j] = 5;break;
                                case 'G':seqlist[i][j] = 6;break;
                                case 'H':seqlist[i][j] = 7;break;
                                case 'I':seqlist[i][j] = 8;break;
                                case 'K':seqlist[i][j] = 9;break;
                                case 'L':seqlist[i][j] = 10;break;
                                case 'M':seqlist[i][j] = 11;break;
                                case 'N':seqlist[i][j] = 12;break;
                                case 'P':seqlist[i][j] = 13;break;
                                case 'Q':seqlist[i][j] = 14;break;
                                case 'R':seqlist[i][j] = 15;break;
                                case 'S':seqlist[i][j] = 16;break;
                                case 'T':seqlist[i][j] = 17;break;
                                case 'V':seqlist[i][j] = 18;break;
                                case 'W':seqlist[i][j] = 19;break;
                                case 'Y':seqlist[i][j] = 20;break;
                                case '*':seqlist[i][j] = 21;break;
                                default: seqlist[i][j] = 0;break;
                                }
                        }
                }
                return seqlist;
}

private double[][] computePDistance(int seqlist[][]){
                double pdistanceMatrix[][] = new double[seqlist.length][seqlist.length];
                for (int i=0;i<seqlist.length;i++){
                        pdistanceMatrix[i][i] = 0;
                        for(int j=i+1;j<seqlist.length;j++){
                                double nl=0;
                                double dif=0;
                                for (int k=0;k<seqlist[1].length;k++){
                                        if ((seqlist[i][k]!=0) && (seqlist[j][k]!=0)){
                                                nl++;
                                                if(seqlist[i][k] != seqlist[j][k]){
                                                        dif++;
                                                }
                                        }
                                }
                                double p = dif/nl;
                                if (nl==0){
                                        p=0;
                                }
                                pdistanceMatrix[i][j] = p;
                                pdistanceMatrix[j][i] = pdistanceMatrix[i][j];
                        }
                }
                return pdistanceMatrix;
}
private double getParalogScore (int seqlist[][],int genomelist[][]){
                double score = 0;
                for (int j=0;j<genomelist[0].length;j++){
                        //12 種間で 1 サイトについてチェック
                        int same = 0;
                        for (int i=0;i<genomelist.length-1;i++){
                                if ((genomelist[i][j]!=0) && (genomelist[i+1][j]!=0)){
```

```
                                                if(genomelist[i][j] == genomelist[i+1][j]){
                                                        same++;
                                                }
                                        }
                                }
                                if (same > genomelist.length-2){
                                        for (int i=0;i<seqlist.length;i++){
                                                if (seqlist[i][j]!=0){
                                                        if(genomelist[0][j] != seqlist[i][j]){
                                                                score++;
                                                        }
                                                }
                                        }
                                }
                        }
                        ///genomelist[0].length
                        return score;
}

private int getStopCodon (int[][] seqmat){
                        int stopcodons = 0;
                        for (int i=0;i<seqmat.length;i++){
                                for (int j=0;j<seqmat[i].length;j++){
                                        if (seqmat[i][j] == 21){
                                                stopcodons++;
                                        }
                                }
                        }
                        return stopcodons;
}

private double testAlignment(double[][] distanceMatrix,double[][] genomeDistance){

                        //12 種の p 平均を計算
                        int n = 0;
                        double psum = 0;
                        for (int i=0;i<genomeDistance.length;i++){
                                for(int j=i+1;j<genomeDistance.length;j++){
                                        psum += genomeDistance[i][j];
                                        n++;
                                }
                        }
                        double refarensAve = psum/n;
                        if (refarensAve < 0.01){
                                refarensAve = 0.01;
                        }
                        System.out.println(refarensAve);

                        double[] scores = new double[distanceMatrix.length];
                        for (int i=0;i<distanceMatrix.length;i++){
                                //対象種の p 平均値を計算
                                n = 0;
                                psum = 0;
                                double targetave = 0;
                                for (int j=0;j<distanceMatrix.length;j++){
                                        psum += distanceMatrix[i][j];
                                        n++;
                                }
                                double targetAve = psum/n;
                                scores[i] = targetAve;
                        }
                        Arrays.sort(scores);
                        System.out.println(scores[scores.length - 1 ]);
                        return scores[scores.length - 1 ] / refarensAve;
}

class ScoreData {
                        private String fastaname;
                        private double score;

                        ScoreData() {
                                // TODO Auto-generated constructor stub
                        }
                        public ScoreData(String fastaname,double score) {
                                this.fastaname = fastaname;
                                this.score = score;
                        }

                        String getName(){
                                return fastaname;
                        }
                        double getScore(){
                                return score;
                        }
                        String getNameScore(){
                                return fastaname + "    score:" + score;
                        }

}

class ScoreComparator implements Comparator<ScoreData> {
```

```
                        public int compare(ScoreData o1, ScoreData o2) {
                                if (o1.getScore() > o2.getScore()){
                                        return 1;
                                }
                                else if (o1.getScore() < o2.getScore()){
                                        return -1;
                                }
                                else{
                                        return 0;
                                }
                        }
                }
                static List<FastaSequence> takeGenomes(List<FastaSequence> fastaList) {
                        List<FastaSequence> genomelist = new ArrayList();
                        String[] genomeName = new String[12];
                        genomeName[0] = "dmel";
                        genomeName[1] = "dsim";
                        genomeName[2] = "dana";
                        genomeName[3] = "dere";
                        genomeName[4] = "dgri";
                        genomeName[5] = "dmoj";
                        genomeName[6] = "dper";
                        genomeName[7] = "dpse";
                        genomeName[8] = "dvil";
                        genomeName[9] = "dwil";
                        genomeName[10] = "dsec";
                        genomeName[11] = "dyak";

                        for (int i=0;i<fastaList.size();i++){
                                String sp = fastaList.get(i).getId();
                                for (int j=0;j<genomeName.length;j++){
                                        if ((sp.indexOf(genomeName[j])!=-1)){
                                                genomelist.add(fastaList.get(i));
                                        }
                                }
                        }
                        return genomelist;
                }
                private List<FastaSequence> removeGenomes(List<FastaSequence> fastaList) {
                        List<FastaSequence> genomelist = new ArrayList();
                        for (int i=0;i<fastaList.size();i++){
                                String sp = fastaList.get(i).getId();
                                if ((sp == "dmel") || (sp == "dsim") || (sp == "dana") || (sp == "dere") || (sp == "dgri") || (sp == "dmoj") || (sp ==
"dmel") || (sp == "per") || (sp == "pse") || (sp == "dvil") || (sp == "dsec") || (sp == "dwil")){

                                }else{
                                        genomelist.add(fastaList.get(i));
                                }
                        }
                        return genomelist;
                }
                private class SwingAction_2 extends AbstractAction {
                        public SwingAction_2() {
                                putValue(NAME, "アミノ酸からパラログチェック");
                                putValue(SHORT_DESCRIPTION, "Some short description");
                        }
                        public void actionPerformed(ActionEvent e) {

                        }
                }
}
```

Concatenator in same folder to one fasta file

```
package practice;
```

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JFileChooser;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.AbstractAction;

import java.awt.event.ActionEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import javax.swing.Action;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;

public class Concatenator extends JFrame {

	private JPanel contentPane;
	private JTextField txtCcdnadatasupermatrix;
	private final Action action = new SwingAction();
	private final Action action_1 = new SwingAction_1();
	private final Action action_2 = new SwingAction_2();
	private JTextArea textArea;
	private JTextArea textArea_1;
	private final Action action_3 = new SwingAction_3();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Concatenator frame = new Concatenator();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Concatenator() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 547, 473);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		txtCcdnadatasupermatrix = new JTextField();
		txtCcdnadatasupermatrix.setText("C:¥¥cDNAData¥¥supermatrix");
		txtCcdnadatasupermatrix.setColumns(10);

		JButton btnNewButton = new JButton("New button");
		btnNewButton.setAction(action);

		JScrollPane scrollPane = new JScrollPane();

		JButton btnNewButton_1 = new JButton("New button");
		btnNewButton_1.setAction(action_1);

		JButton button = new JButton("¥u672B¥u7AEF¥u3092¥u30AB¥u30C3¥u30C8");
		button.setAction(action_2);

		JScrollPane scrollPane_1 = new JScrollPane();

		JButton button_1 = new JButton("90%¥u306E¥u30B5¥u30A4¥u30C8¥u306E¥u307F¥u3092¥u62BD¥u51FA");
		button_1.setAction(action_3);
		GroupLayout gl_contentPane = new GroupLayout(contentPane);
		gl_contentPane.setHorizontalGroup(
			gl_contentPane.createParallelGroup(Alignment.TRAILING)
```

```
                                        .addGroup(gl_contentPane.createSequentialGroup()
                                                .addContainerGap()
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.TRAILING)
                                                        .addComponent(scrollPane,      GroupLayout.DEFAULT_SIZE,
381, Short.MAX_VALUE)
                                                        .addComponent(scrollPane_1,                Alignment.LEADING,
GroupLayout.DEFAULT_SIZE, 381, Short.MAX_VALUE)
                                                        .addGroup(gl_contentPane.createSequentialGroup()
                                                                .addComponent(txtCcdnadatasupermatrix,
GroupLayout.DEFAULT_SIZE, 312, Short.MAX_VALUE)

                .addPreferredGap(ComponentPlacement.UNRELATED)
                                                                .addComponent(btnNewButton))
                                                        .addGroup(Alignment.LEADING,
gl_contentPane.createSequentialGroup()
                                                                .addComponent(btnNewButton_1,
GroupLayout.PREFERRED_SIZE, 159, GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(ComponentPlacement.RELATED)
                                                                .addComponent(button,
GroupLayout.PREFERRED_SIZE, 146, GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(ComponentPlacement.RELATED)
                                                                .addComponent(button_1)))
                                                .addContainerGap())
                        );
                        gl_contentPane.setVerticalGroup(
                                gl_contentPane.createParallelGroup(Alignment.LEADING)
                                        .addGroup(gl_contentPane.createSequentialGroup()
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(txtCcdnadatasupermatrix,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(btnNewButton))
                                                .addGap(8)
                                                .addComponent(scrollPane,       GroupLayout.PREFERRED_SIZE,       179,
GroupLayout.PREFERRED_SIZE)
                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                .addComponent(scrollPane_1,       GroupLayout.DEFAULT_SIZE,       174,
Short.MAX_VALUE)
                                                .addPreferredGap(ComponentPlacement.RELATED)
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(btnNewButton_1)
                                                        .addComponent(button)
                                                        .addComponent(button_1))
                                                .addContainerGap())
                        );

                        textArea_1 = new JTextArea();
                        scrollPane_1.setViewportView(textArea_1);

                        textArea = new JTextArea();
                        scrollPane.setViewportView(textArea);
                        contentPane.setLayout(gl_contentPane);
                }
                private class SwingAction extends AbstractAction {
                        public SwingAction() {
                                putValue(NAME, "参照");
                                putValue(SHORT_DESCRIPTION, "Some short description");
                        }
                        public void actionPerformed(ActionEvent e) {
                                JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
                                filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                                if (selected == JFileChooser.APPROVE_OPTION){
                                        txtCcdnadatasupermatrix.setText(filechooser.getSelectedFile().getAbsolutePath());
                                        File file = new File(txtCcdnadatasupermatrix.getText());
                                        String[] filelist = file.list();
                                }
                        }
                }
                private class SwingAction_1 extends AbstractAction {
                        public SwingAction_1() {
                                putValue(NAME, "スーパーマトリックス作成");
                                putValue(SHORT_DESCRIPTION, "Some short description");
                        }
                        public void actionPerformed(ActionEvent e) {
                                String filead = txtCcdnadatasupermatrix.getText();
                                try {
                                        concatenate(filead);
                                } catch (IOException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                }

                        }
                }
                private void concatenate(String filead) throws IOException {
                        File file = new File(filead);
                        setTitle(file.getAbsolutePath());
                        String fileAdress = file.getAbsolutePath();
```

```
                String fileParentAdress = file.getParent();
                String ntlist[] = file.list(new FastaNameFilter());

                //種数だけ配列をつくる
                List<FastaSequence> first = SequenceUtil.readFasta(new FileInputStream(fileAdress +"¥¥"+ ntlist[0]));
                Collections.sort(first);
        int numsp = first.size();
                String[] seq = new String[numsp];
                String[] name = new String[numsp];
                ArrayList<String> spnamelist = new ArrayList();
                for (int i=0;i<numsp;i++){
                        spnamelist.add(first.get(i).getId().substring(0,4));
                }

                //遺伝子ごとに処理
                String geneNumbers = "";
                for (int i=0;i<ntlist.length;i++){
                        String infilename = fileAdress +"¥¥"+ ntlist[i];
            FileInputStream input = new FileInputStream(infilename);
                        List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
                        input.close();
                        if (fastaList.size() < numsp){
                                System.out.println("1 種欠け");
                                for (int j=0;j<numsp;j++){
                                        for (int k=0;k<fastaList.size();k++){
                                                if (fastaList.get(k).getId().indexOf(spnamelist.get(j))!= -1){
                                                        break;
                                                }else if (k == fastaList.size()-1){
                                                        System.out.println("修正");
                                                        int length = fastaList.get(0).getLength();
                                                        String nullseq = "";
                                                        for (int l=0;l<length;l++){
                                                                nullseq += "-";
                                                        }
                                                        fastaList.add(new
FastaSequence(spnamelist.get(j),nullseq));
                                                        break;
                                                }
                                        }
                                }
                        }
                        System.out.println(ntlist[i]);
                        Collections.sort(fastaList);;
                        //種ごとに処理
                        for (int j=0;j<fastaList.size();j++){
                                String repair = fastaList.get(j).getSequence().toString();
                                if (i==0){
                                        seq[j]=repair;
                                }
                                else{
                                        seq[j]    += repair;
                                        name[j] = fastaList.get(j).getId();
                                }
                        }
                        String repair = fastaList.get(0).getSequence().toString();
                        int start = seq[0].length() - repair.length()+1;
                        int end = seq[0].length();
                        String geneSite = start + "-" + end;
                        //遺伝子ごとの長さを表示
                        //textArea.append(ntlist[i] + " : sequence length : " + repair.length() + "¥n");
                        textArea.append(ntlist[i] + " = " + start + "-" + end + ";¥n");
                        //textArea.append("charset " + ntlist[i] + " = " + geneSite + ";¥n");
                        for (int j=0;j<3;j++){
                                int startMod = start + j;
                                textArea_1.append(ntlist[i] + "_pos" + j + " = " + startMod + "-" + end + "¥¥3;¥n");
                        }
                        geneNumbers += " " + ntlist[i] + ",";
                }
                List<FastaSequence> writeList = new ArrayList<FastaSequence>();
        for(int i=0;i<seq.length;i++){
                FastaSequence proFs = new FastaSequence(name[i].substring(0,4),seq[i]);
                writeList.add(proFs);
        }
        textArea.append("partition favored = " + ntlist.length +":"+ geneNumbers.substring(0,geneNumbers.length()-1) + ";");
        String outfilename = filead +"¥¥superMatrix.fasta";
                OutputStream os = new FileOutputStream(outfilename);
        SequenceUtil.writeFasta(os, writeList);
                os.close();
    }
    private class SwingAction_2 extends AbstractAction {
                public SwingAction_2() {
                        putValue(NAME, "末端をカット");
                        putValue(SHORT_DESCRIPTION, "Some short description");
                }
                public void actionPerformed(ActionEvent e) {
                        String filead = txtCcdnadatasupermatrix.getText();
                        try {
                                modTerminal(filead);
                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
```

```
                                        e1.printStackTrace();
                                }
                        }
}
private List<FastaSequence> cutTerminal(List<FastaSequence> fastaList){
                        List<FastaSequence> preList = fastaList;
                        List<FastaSequence> genomeList = DistanceMatrixMaker.takeGenomes(fastaList);
                        int start[] = new int[genomeList.size()];
                        int end[] = new int[genomeList.size()];
                        for (int i=0;i<genomeList.size();i++){
                                String seq = genomeList.get(i).getSequence();
                                for (int j=0;j<seq.length();j++){
                                        if ((seq.charAt(j) != '-') && (seq.charAt(j) != 'X')){
                                                start[i] = j;
                                                break;
                                        }
                                }
                                for (int j=seq.length()-1;j > 0 ;j--){
                                        if ((seq.charAt(j) != '-') && (seq.charAt(j) != 'X')) {
                                                end[i] = j;
                                                break;
                                        }
                                }
                        }
                        int ss = getTopShare(start);
                        int es = getTopShare(end);
                        List<FastaSequence> modlist = new ArrayList();
                        for (int j=0;j<preList.size();j++){
                                String repair = preList.get(j).getSequence().toString().substring(ss,es+1);
                                modlist.add(new FastaSequence(preList.get(j).getId(),repair));
                        }
                        return modlist;
}
private int getTopShare(int[] start){
                        ArrayList sitelist = new ArrayList();
                        sitelist.add(start[0]);
                        for (int i=0;i<start.length;i++){
                                for (int j=i+1;j<start.length;j++){
                                        if (start[i]!=start[j]){
                                                sitelist.add(start[j]);
                                        }
                                }
                        }
                        ArrayList scoreList = new ArrayList();
                        for (int i=0;i<sitelist.size();i++){
                                int score = 0;
                                for (int j=0;j<start.length;j++){
                                        if ((int)sitelist.get(i) == start[j]){
                                                score++;
                                        }
                                }
                                scoreList.add(score);
                        }
                        int topscore = 0;
                        for (int i=0;i<scoreList.size();i++){
                                for (int j=i+1;j<scoreList.size();j++){
                                        if ((int)scoreList.get(i) < (int)scoreList.get(j)){
                                                topscore = (int)scoreList.get(j);
                                        }
                                }
                        }
                        return (int)sitelist.get(topscore);
                        /*
                        int common[] = new int[start.length];
                        for (int i=0;i<start.length;i++){
                                int share =0;
                                for(int j=i+1;j<start.length;j++){
                                        if (start[i]==start[j]){
                                                share++;
                                        }
                                        common[i] = share;
                                }
                        }
                        int top = 0;
                        for (int i=0;i<start.length-1;i++){
                                if (common[i] < common[i+1]){
                                        top = i+1;
                                }
                        }
                        return start[top];*/
}

private void modTerminal(String filead) throws IOException{
                        File file = new File(filead);
                        setTitle(file.getAbsolutePath());
                        String fileAdress = file.getAbsolutePath();
                        String fileParentAdress = file.getParent();
                        String ntlist[] = file.list(new FastaNameFilter());

                        for (int i=0;i<ntlist.length;i++){
                                String infilename = fileAdress +"¥¥"+ ntlist[i];
```

```
                        FileInputStream input = new FileInputStream(infilename);
                                List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
                                input.close();
                                fastaList = cutTerminal(fastaList);
                        String outfilename = filead +"¥¥mod"+ntlist[i];
                                OutputStream os = new FileOutputStream(outfilename);
                        SequenceUtil.writeFasta(os, fastaList);
                                os.close();
                        }
                }
                private class SwingAction_3 extends AbstractAction {
                        public SwingAction_3() {
                                putValue(NAME, "90%カバーサイトの抽出");
                                putValue(SHORT_DESCRIPTION, "Some short description");
                        }
                        public void actionPerformed(ActionEvent e) {

                        }
                }
                private void getCov90(List<FastaSequence> fastalist){

                }
}
```

GC counter

```
package practice;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
```

```
import javax.swing.GroupLayout.Alignment;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JFileChooser;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.AbstractAction;

import java.awt.event.ActionEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

import javax.swing.Action;

import org.biojava3.data.sequence.FastaSequence;
import org.biojava3.data.sequence.SequenceUtil;
import org.biojava3.core.sequence.*;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class GCcounter extends JFrame {

	private JPanel contentPane;
	private JTextField textField;
	private final Action action = new SwingAction();
	private final Action action_1 = new SwingAction_1();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					GCcounter frame = new GCcounter();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public GCcounter() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		textField = new JTextField("C:¥¥cDNAData¥¥supermatrix");
		textField.setColumns(10);

		JButton btnBrowse = new JButton("browse");
		btnBrowse.setAction(action);

		JScrollPane scrollPane = new JScrollPane();

		JButton btnStart = new JButton("start");
		btnStart.setAction(action_1);
		GroupLayout gl_contentPane = new GroupLayout(contentPane);
		gl_contentPane.setHorizontalGroup(
			gl_contentPane.createParallelGroup(Alignment.LEADING)
				.addGroup(Alignment.TRAILING, gl_contentPane.createSequentialGroup()
					.addComponent(textField, GroupLayout.DEFAULT_SIZE, 348,
Short.MAX_VALUE)
					.addPreferredGap(ComponentPlacement.RELATED)
					.addComponent(btnBrowse, GroupLayout.DEFAULT_SIZE,
GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
				.addComponent(scrollPane, Alignment.TRAILING, GroupLayout.DEFAULT_SIZE, 436,
Short.MAX_VALUE)
				.addGroup(Alignment.TRAILING, gl_contentPane.createSequentialGroup()
					.addGap(175)
					.addComponent(btnStart, GroupLayout.DEFAULT_SIZE, 91,
Short.MAX_VALUE)
					.addGap(158))
		);
		gl_contentPane.setVerticalGroup(
			gl_contentPane.createParallelGroup(Alignment.LEADING)
				.addGroup(gl_contentPane.createSequentialGroup()
```

```
                                                .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
                                                        .addComponent(textField,   GroupLayout.PREFERRED_SIZE,
GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                                                        .addComponent(btnBrowse))
                                                .addGap(8)
                                                .addComponent(scrollPane,        GroupLayout.PREFERRED_SIZE,        193,
GroupLayout.PREFERRED_SIZE)
                                                .addPreferredGap(ComponentPlacement.RELATED, 9, Short.MAX_VALUE)
                                                .addComponent(btnStart))
                        );

                        JTextArea textArea = new JTextArea();
                        scrollPane.setViewportView(textArea);
                        contentPane.setLayout(gl_contentPane);
                }
                private class SwingAction extends AbstractAction {
                        public SwingAction() {
                                putValue(NAME, "Browse");
                                putValue(SHORT_DESCRIPTION, "Some short description");
                        }
                        public void actionPerformed(ActionEvent e)    {
                                JFileChooser filechooser = new JFileChooser("C:¥¥cDNAData");
                                filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                                int selected = filechooser.showOpenDialog(contentPane); //「開く」ダイアログ表示
                                if (selected == JFileChooser.APPROVE_OPTION){
                                        textField.setText(filechooser.getSelectedFile().getAbsolutePath());
                                }
                        }
                }
                private class SwingAction_1 extends AbstractAction {
                        public SwingAction_1() {
                                putValue(NAME, "GCcount");
                                putValue(SHORT_DESCRIPTION, "Some short description");
                        }
                        public void actionPerformed(ActionEvent e) {
                                try {
                                        countGC(textField.getText());
                                } catch (IOException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                }
                        }
                }
                private void countGC(String filead) throws IOException {
                        File file = new File(filead);
                        setTitle(file.getAbsolutePath());
                        String fileAdress = file.getAbsolutePath();
                        String fileParentAdress = file.getParent();
                        String ntlist[] = file.list(new FastaNameFilter());
                        Workbook wb = new XSSFWorkbook();
                        Sheet sheet = wb.createSheet();
                        Row labelrow = sheet.createRow(0);

                        for (int i=0;i<ntlist.length;i++){
                                String infilename = fileAdress +"¥¥"+ ntlist[i];
                    FileInputStream input = new FileInputStream(infilename);
                                List<FastaSequence> fastaList = SequenceUtil.readFasta(input);
                                input.close();
                                Row row = sheet.createRow(i+1);

                                for (int j=0;j<fastaList.size();j++){
                                        String seq = fastaList.get(j).getSequence().replaceAll("-", "");
                                        DNASequence ds = new DNASequence(seq);
                                        double gcPercent = ((double)ds.getGCCount())/((double)ds.getLength())*100;
                                        if (i==0){
                                                Cell labelcell = labelrow.createCell(j+1);
                                                labelcell.setCellValue(fastaList.get(j).getId());
                                        }
                                        if (j==0){
                                                Cell gen = row.createCell(0);
                                                gen.setCellValue(ntlist[i]);
                                        }
                                        Cell cell = row.createCell(j+1);
                                        cell.setCellValue(gcPercent);
                                }
                                FileOutputStream out = null;
                            try{
                               out = new FileOutputStream(fileAdress + "¥¥GCpercent.xlsx");
                               wb.write(out);
                            }catch(IOException e){
                               System.out.println(e.toString());
                            }finally{
                               try {
                                  out.close();
                               }catch(IOException e){
                                  System.out.println(e.toString());
                               }
                            }
                        }
                }
}
```